July 2019

Sparsity in Machine Learning: An Information Selecting Perspective

Siwei Feng
Electrical and Computer Engineering

SPARSITY IN MACHINE LEARNING: AN INFORMATION SELECTING PERSPECTIVE

A Dissertation Outline Presented

by

SIWEI FENG

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

May 2019

Electrical and Computer Engineering

SPARSITY IN MACHINE LEARNING: AN INFORMATION SELECTING PERSPECTIVE

A Dissertation Outline Presented

by

SIWEI FENG

ABSTRACT

SPARSITY IN MACHINE LEARNING: AN INFORMATION SELECTING PERSPECTIVE

MAY 2019

SIWEI FENG

B.Sc., SOOCHOW UNIVERSITY

M.Sc., UNIVERSITY OF MASSACHUSETTS AMHERST

Ph.D., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor Marco F. Duarte

Today we are living in a world awash with data. Large volumes of data are acquired, analyzed and applied to tasks through machine learning algorithms in nearly every area of science, business, and industry. For example, medical scientists analyze the gene expression data from a single specimen to learn the underlying causes of disease (e.g. cancer) and choose the best treatment; retailers can know more about customers' shopping habits from retail data to adjust their business strategies to better appeal to customers; suppliers can enhance supply chain success through supply chain systems built on knowledge sharing. However, it is also reasonable to doubt whether all the genes make contributions to a disease; whether all the data obtained from existing customers can be applied to a new customer; whether all shared knowledge in the supply network is useful to a specific supply scenario. Therefore, it is crucial to sort through the massive information provided by data and keep what we really need. This process is referred to as information selection, which keeps the information that helps improve the performance of corresponding machine learning tasks and discards information that is useless or even harmful to task performance. Sparse learning is a powerful tool to achieve information selection. In this thesis, we apply sparse learning to two major areas in machine learning – feature selection and transfer learning.

Feature selection is a dimensionality reduction technique that selects a subset of representative features. Recently, feature selection combined with sparse learning has attracted significant attention due to its outstanding performance compared with traditional feature selection methods that ignore correlation between features. However, they are restricted by design to linear data transformations, a potential drawback given that the underlying correlation structures of data are often non-linear. To leverage more sophisticated embedding than the linear model assumed by sparse learning, we propose an autoencoder-based unsupervised feature selection approach that leverages a single-layer autoencoder for a joint framework of feature selection and manifold learning. Additionally, we include spectral graph analysis on the projected data into the learning process to achieve local data geometry preservation from the original data space to the low-dimensional feature space.

Transfer learning describes a set of methods that aim at transferring knowledge from related domains to alleviate the problems caused by limited/no labeled training data in machine learnig tasks. Many transfer learning techniques have been proposed to deal with different application scenarios. However, due to the differences in data distribution, feature space, label space, etc., between source domain and target domain, it is necessary to select and only transfer relevant information from source domain to improve the performance of target learner. Otherwise, the target learner can be negatively impacted by the weak-related knowledge from source domain, which is referred to as negative transfer. In this thesis, we focus on two transfer learning scenarios for which limited labeled training data are available in target domain. In the first scenario, no label information is avaible in source data. In the second scenario, large amounts of labeled source data are available, but there is no overlap between the source and target label spaces. The corresponding transfer learning technique to the former case is called *self-taught learning*, while that for the latter case is called *few-shot learning*. We apply self-taught learning to visual, textal, and audio data. We also apply few-shot learning to wearable sensor based human activity data. For both cases, we propose a metric for the relevance between a target sample/class and a source sample/class, and then extract information from the related samples/classes for knowledge transfer to perform information selection so that negative transfer caused by weakly related source information can be alleviated. Experimental results show that transfer learning can provide better performance with information selection.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

Machine learning practitioners are always yearning for training data with large volume and variety since it is believed that those can help train models with more useful information characterized and thus leading to better task performance. However, in practice a large dataset is not always a guarantee for good machine learning task performance since there may exist samples and/or features that are redundant or irrelevant to specific machine learning tasks. In other words, models learned with these data may contain useless or even harmful information that negatively affect task performance. To alleviate the negative influence brought by such "bad" information, feature/sample selection is a necessary preprocessing step. Within the significant literatures on feature/sample selection, sparse learning based approaches play an important role due to the selecting nature of sparse learning. In this thesis, sparse learning in unsupervised feature selection and transfer learning is investigated.

1.1 Unsupervised Feature Selection

In recent years, high-dimensional data can be found in many areas such as computer vision, pattern recognition, data mining, etc. Hign dimensionality enables data to include more information. However, learning high-dimensional data often suffer from several issues. For example, with a fixed number of training data, a large data dimensionality can cause the so-called Hughes phenomenon, i.e., a reduction in the generalization of the learned models due to overfitting during the training procedure compared with lower dimensional data [1]. Moreover, high-dimensional data tend to include significant redundancy in adjacent features, or even noise, which leads to large amounts of useless or even harmful information being processed, stored, and transmitted [2, 3]. All these issues present challenges to many conventional data analysis problems. Moreover, several papers in the literature have shown that the intrinsic dimensionality of high-dimensional data is actually small [4–6]. Thus, dimensionality reduction is a popular preprocessing step for high-dimensional data analysis, which decreases time for data processing and also improves generalization of learned models.

Feature selection [7–12] is a set of frequently used dimensionality reduction approaches that aim at selecting a subset of input dimensions[1]. Feature selection has the advantage of preserving the same feature space as that of raw data. Feature selection methods can be categorized into groups based on different criteria summarized below; refer to [13] for a detailed survey on feature selection.

- **Label Availability.** Based on the availability of label information, feature selection algorithms can be classified into supervised [7–9], semi-supervised [10–12], and unsupervised [14–30] methods. Since labeled data are usually expensive and time-consuming to acquire [31, 32], unsupervised feature selection has been gaining more and more attention recently and is the subject of our focus in this work.

- **Search Strategy.** In terms of selection strategies, feature selection methods can be categorized into filter, wrapper, and embedded methods. Wrapper methods [33, 34] are seldom used in practice since they rely on a repetition of feature subset searching and selected feature subset evaluation until some stopping criteria or some desired performance are reached, which requires an exponential search space and thus is computationally prohibitive when feature dimensionality is high. Filter feature selection methods, e.g. Laplacian score

[1]In the sequel, we use “feature” to replace “input dimension” for convenience in description.

(LapScore) [14] and SPEC [15], assign a score (measuring task relevance, redundancy, etc.) to each feature and select those with the best scores. Though convenient to computation, these methods are often tailored specifically for a given task and may not provide an appropriate match to the specific application of interest. Embedded methods combine feature selection and model learning and provide a compromise between the two earlier extremes, as they are more efficient than wrapper methods and more task-specific than filter methods. We focus on embedded feature selection methods.

In recent years, feature selection algorithms aiming at selecting features that preserve intrinsic data structure (such as subspace or manifold structure) [16–30] have attracted significant attention due to their good performance and interpretability [13]. In these methods, data are linearly projected onto new spaces through a transformation matrix, with fitting errors being minimized along with some sparse regularization terms. Feature importance is usually scored using the norms of corresponding rows/columns in the transformation matrix. In some methods [20–25, 28–30], the local data geometric structure, which is usually characterized by nearest neighbor graphs, is also preserved in the low-dimensional projection space. However, one basic assumption of these methods is that the data to be processed lie in or near a completely linear low-dimensional manifold. However, this is not always true in practice, in particular with more sophisticated data.

We propose a novel algorithm for graph and autoencoder-based feature selection (GAFS). In this method, we integrate three objective functions into a single optimization framework: (*i*) we use a single-layer autoencoder to reconstruct the input data; (*ii*) we use an $\ell_{2,1}$-norm penalty on the columns of the weight matrix connecting the autoencoder's input layer and hidden layer to provide feature selection; (*iii*) we preserve the local geometric structure of the data through to the corresponding hidden layer activations. Extensive experiments are conducted on image data, audio data, text data, and biological data. Many experimental results are provided to demonstrate the outstanding performance achieved by the proposed method compared with other state-of-the-art unsupervised feature selection algorithms.

The key contributions are highlighted as follows.

- We propose a novel unsupervised feature selection framework which is based on an autoencoder and graph data regularization. By using this framework, the information of the underlying data subspace can be leveraged, which loosens the assumption of linear manifold in many relevant techniques.

- We present an efficient solver for the optimization problem underlying the proposed unsupervised feature selection scheme. Our approach relies on an iterative scheme based on the gradient descent of the proposed objective function.

- We provide multiple numerical experiments that showcase the advantages of the flexible models used in our feature selection approach with respect to the state-of-the-art approaches from the literature.

1.2 Transfer Learning

Supervised learning has excelled in many machine learning tasks such as classification [3, 35] and regression [36, 37]. However, the success of a supervised learning algorithm requires large-scale labeled training datasets and that both training and testing data sharing the same label and feature space.[2] These conditions limit the applications of supervised learning methods in practical scenarios since it is expensive to collect eligible training data [38, 39].

Several techniques have been proposed to tackle the limitations of supervised learning methods. Semi-supervised learning [40–42] algorithms use both labeled and

[2] "Training data" is used in the sequel to denote data used for model learning.

unlabeled data to improve performance when labeled training data are limited. However, many semi-supervised learning algorithms assume that unlabeled data and labeled data have the same distribution [40, 41] or class labels [42]. The success of semi-supervised learning highly depends on the validity of these assumptions. However, it is still difficult to gather unlabeled data which satisfy these preconditions.

In order to further loosen the restrictions on training data, many transfer learning approaches [43, 44] have been proposed. Transfer learning methods use the knowledge obtained from a source domain to improve the performance on target domain tasks. We mainly focus on two transfer learning techniques: self-taught learning and few-shot learning.

1.2.1 Self-Taught Learning

Self-taught learning [45–53] is the type of transfer learning techniques most similar to semi-supervised learning, which also employs unlabeled data with the attempt to improve supervised learning performance when labeled training data are limited. However, compared with semi-supervised learning, self-taught learning methods have fewer restrictions on unlabeled data, as they allow the label spaces and marginal probability distributions of unlabeled and labeled data to be different. In self-taught learning, unlabeled data are used as source from which the knowledge learned is applied to tasks performed on labeled target data. Such a loose restriction on unlabeled data significantly simplifies learning due to the huge volume of unlabeled data we can access. However, the easily obtained unlabeled data inevitably contain samples with weak relation to the labeled training data, which may even harm the supervised learning performance if we treat them equally as other unlabeled samples during knowledge transfer. This is known as negative transfer. Therefore, it is necessary to select samples that are related to the labeled data to reduce the impact caused by negative transfer.

We propose a novel algorithm for self-taught learning with unlabeled source data which are related to labeled target data to be selected. The algorithm leverages a linear mapping, a k-nearest neighbor (kNN) graph and a single-layer autoencoder to obtain a metric for cross domain sample relevance. We refer to this method as graph and autoencoder-based self-taught learning (GASTL). The framework of GASTL includes two modules: a source sample re-weighting module and a classifier training module. In the first module, we assign each unlabeled source sample a weight that indicates its relevance to labeled target samples.[3] In the second module, source samples with large weights are selected to combine a training set with target data to train a classifier. Each selected source sample is assigned a pseudo-label from the target domain label space to be used during classifier training. The weights of source samples are also used during classifier training. The trained classifier is then used to predict labels of unseen target samples.

The key contributions are as follows:

- We propose a novel metric for the relevance of each source sample to the target domain in the scenario of self-taught learning based on an autoencoder and graph data regularization. To the best of our knowledge, we are the first to measure source and target sample relevance for self-taught learning problems.

- We propose a novel classifier training scheme with both selected source samples and target samples as training dataset with the relevance of each source sample to target domain being considered. We are not aware of existing self-taught learning approaches that integrate cross domain sample relevance into classifier training.

[3]The setting of self-taught learning requires source samples to be unlabeled and target samples to be labeled. Therefore in the sequel we do not specify the availability of label information for both source and target samples.

- We present an efficient solver for the knowledge transfer optimization problem described above that relies on an iterative scheme based on the gradient descent of the proposed objective function. This solver shows advantages when the model complexity is large.

- Multiple experimental results are provided to demonstrate the performance improvements in terms of classification accuracy and insensitivity to parameters achieved by the proposed method compared with state-of-the-art self-taught learning methods and other relevant techniques.

1.2.2 Few-Shot Learning

Few-shot learning [54–56] assumes availability of labeled training data in the target domain as in self-taught learning. Unlike self-taught learning, few-shot learning problems require large amounts of labeled source data whose label spaces have no overlap with that of target domain. Negative transfer also needs to be alleviated in this scenario.

We do not perform few-shot learning on multiple types of data as what we do with self-taught learning. Instead, we focus on human activity recognition with data obtained from wearable sensors such as a smartphone or wristband, which predicts activity type (e.g. walking, swimming, etc.) from the sensor outputs. Human activity recognition has been applied in many tasks such as sleep state detection [57] and smart home sensing [58].

In order to have a better discrimination of different types of activities, models need to be trained with large amounts of data from a diversity of sources. Unfortunately, in practice we do not always have enough data for each activity type. For example, a wearable health care system needs to be retrained if the previously collected activity data are not representative of the new activity type, and it is not likely for users to provide large amount of data for a single activity type. However, there may exist relevance between new and existing activity data. Therefore, relevant knowledge from exisiting activity data can be used in model training with new data.

The algorithm we propose to perform few-shot learning on human activity recognition leverages a deep neural network and a linear mapping to obtain a metric for cross domain class relevance. The framework includes a feature extraction module and source class weighting module. In the first module, we use the a neural network to extract features for both source and target samples. In the second module, cross domain class similarities are measured with the features obtained in the first module and parameters of relevant source classes are used for target domain classifier training. The trained classifier is then used to predict the activity type of unseen target samples.

1.3 Outline

The thesis is organized as follows.

Chapter 2 provides notations for this thesis, the concept of single-layer autoencoder which plays an important role in algorithms described in Chapter 3 and Chapter 4, and a brief literature review of sparse learning-based unsupervised feature selection, self-taught learning, and few-shot learning. Chapter 3 presents our proposed unsupervised feature selection approach. Chapter 4 presents our proposed self-taught learning approach. Chap 5 presents our proposed few-shot learning approach for human activity recognition. Finally, we conclude with a summary of our findings and a discussion of ongoing work in Chapter 6.

CHAPTER 2
BACKGROUND

2.1 Notations

Vectors are denoted by bold lowercase letters while matrices are denoted by bold uppercase letters. The superscript T of a matrix denotes the transposition operation. For a matrix $\mathbf{A}$, $\mathbf{A}^{(q)}$ denotes the q^{th} column and $\mathbf{A}_{(p)}$ denotes the p^{th} row, while $\mathbf{A}^{(p,q)}$ denotes the entry at the p^{th} row and q^{th} column. The $\ell_{r,p}$-norm for a matrix $\mathbf{W} \in \mathbb{R}^{a \times b}$ is denoted as

$$\|\mathbf{W}\|_{r,p} = \left(\sum_{i=1}^{a} \left(\sum_{j=1}^{b} |\mathbf{W}^{(i,j)}|^{r} \right)^{p/r} \right)^{1/p} . \tag{2.1}$$

The trace of a matrix $\mathbf{L} \in \mathbb{R}^{a \times a}$ is defined as $\text{Tr}(\mathbf{L}) = \sum_{i=1}^{a} \mathbf{L}^{(i,i)}$. We use $\mathbf{1}$ and $\mathbf{0}$ to denote an all-ones and all-zeros matrix or vector of the appropriate size, respectively.

We use $\mathbf{X} = [\mathbf{X}^{(1)}, \mathbf{X}^{(2)}, \cdots, \mathbf{X}^{(n)}] \in \mathbb{R}^{d \times n}$ to denote sample sets, where $\mathbf{X}^{(i)} \in \mathbb{R}^{d}$ is the i^{th} sample in $\mathbf{X}$ for $i = 1, 2, \cdots, n$, and where d and n denote data dimensionality and number of samples in $\mathbf{X}$, respectively.

For notations in transfer learning, we use $\mathcal{D}$ to denote a domain and $\mathcal{T}$ for a task. A domain $\mathcal{D}$ consists of a feature space $\mathcal{X}$ and a marginal probability distribution $P(\mathbf{X})$ over a sample set $\mathbf{X}$. A task $\mathcal{T}$ consists of a label space $\mathcal{Y}$ and an objective predictive function $f(\mathbf{X}, \mathbf{Y})$ to predict the corresponding labels $\mathbf{Y}$ of a sample set $\mathbf{X}$. We use $\mathcal{D}_{\text{src}} = \{\mathcal{X}_{\text{src}}, P(\mathbf{X}_{\text{src}})\}$ and $\mathcal{T}_{\text{src}} = \{\mathcal{Y}_{\text{src}}, f(\mathbf{X}_{\text{src}}, \mathbf{Y}_{\text{src}})\}$ to denote the source domain and task, and use $\mathcal{D}_{\text{trg}} = \{\mathcal{X}_{\text{trg}}, P(\mathbf{X}_{\text{trg}})\}$ and $\mathcal{T}_{\text{trg}} = \{\mathcal{Y}_{\text{trg}}, f(\mathbf{X}_{\text{trg}}, \mathbf{Y}_{\text{trg}})\}$ for the target domain and task.

2.2 Single-Layer Autoencoder

A single-layer autoencoder is an artificial neural network that aims to learn a function $h(\mathbf{x}; \mathbf{\Theta}) \approx \mathbf{x}$ with a single hidden layer, where $\mathbf{x} \in \mathbb{R}^d$ is the input data, $h(\cdot)$ is a nonlinear function, and $\mathbf{\Theta}$ is a set of parameters. To be more specific, the workflow of an autoencoder contains two steps:

- Encoding: mapping the input data $\mathbf{x}$ to a compressed data representation $\mathbf{y} \in \mathbb{R}^m$:

  $$\mathbf{y} = \sigma(\mathbf{W_1}\mathbf{x} + \mathbf{b}_1), \tag{2.2}$$

  where $\mathbf{W}_1 \in \mathbb{R}^{m \times d}$ is a weight matrix, $\mathbf{b}_1 \in \mathbb{R}^m$ is a bias vector, and $\sigma(\cdot)$ is an elementary nonlinear activation function. Commonly used activation functions include the sigmoid function, the hyperbolic tangent function, the rectified linear unit, etc.

- Decoding: mapping the compressed data representation $\mathbf{y}$ to a vector in the original data space $\bar{\mathbf{X}} \in \mathbb{R}^{\mathbf{d}}$:

  $$\bar{\mathbf{X}} = \sigma(\mathbf{W_2}\mathbf{y} + \mathbf{b}_2), \tag{2.3}$$

  where $\mathbf{W}_2 \in \mathbb{R}^{d \times m}$ and $\mathbf{b}_2 \in \mathbb{R}^d$ are the corresponding weight matrix and bias vector, respectively.

The optimization problem brought by the autoencoder is to minimize the difference between the input data and the reconstructed/output data. To be more specific, given a set of data $\mathbf{X} = [\mathbf{X}^{(1)}, \mathbf{X}^{(2)}, \cdots, \mathbf{X}^{(n)}]$, the parameters $\mathbf{W}_1$, $\mathbf{W}_2$, $\mathbf{b}_1$, and $\mathbf{b}_2$ are adapted to minimize the reconstruction error $\sum_{i=1}^{n} \|\mathbf{X}^{(i)} - \bar{\mathbf{X}}^{(i)}\|_2^2$, where $\bar{\mathbf{X}}^{(i)}$ is the output of autoencoder to the input $\mathbf{X}^{(i)}$. The general approach to minimize the reconstruction error is by selecting the parameter values via the backpropagation algorithm.[1]

The data reconstruction capability of the autoencoder makes it suitable to capture the essential information of the data while discarding information that is not useful or redundant.

2.3 Long-Short Term Memory Network

A long-short term memory (LSTM) network [60] is a type of recurrent neural network (RNN) which processes time series signals by taking as their input not just the current inputs but also what they have processed earlier in time. Each RNN contains a loop (repeating modules) inside the network structure that allows information to be passed from one step of the network to the next.

RNNs show their limitations when long-term dependencies are needed to be captured. Consider the task of predicting the last word in the text "I come from Jiangsu, a province in China. The closest noun "province" suggests that the last word is probably the name of a country, but if we want to know which country, we need information from further back. When the gap between the position of relevant information and the point where it is needed becomes large, RNNs show their incapabilities to connect in practice.

LSTMs are a special kind of RNN, which are famous for their capabilities to capture long-term dependencies. Compared with the simple repeating modules of most RNNs, which sometimes only contans a single tanh layer, the repeating modules of LSTMs include more complicated interacting layers in their structures. The workflow of LSTM can be briefly described as follows. The first step is to determine the importance of previous information, which is to decide a status between "completely

[1]For more details on the backpropagation algorithm, refer to the survey paper [59].

forget about this" and "completely keep this". The next step is to decide what new information to store in the cell state and then replace the old state with a new state. Finally, we decide the information to output.

A stacked LSTM model [61] is a LSTM model with multiple amultiple hidden LSTM layers where each layer contains multiple memory cells. By stacking LSTM hidden layers, a LSTM model can be deeper that makes it capable of tackling more complex problems.

2.4 Sparse Learning-Based Unsupervised Feature Selection

Many unsupervised feature selection methods based on subspace structure preservation have been proposed in the past decades. For cases missing class labels, unsupervised feature selection methods select features that are representative of the underlying subspace structure of the data [16]. The basic idea is to use a transformation matrix to project data to a new space and guide feature selection based on the sparsity of the transformation matrix [17]. To be more specific, the generic framework of these methods is based on the optimization

$$\min_{\mathbf{W}} \mathcal{L}(\mathbf{Y}, \mathbf{W}\mathbf{X}) + \lambda \mathcal{R}(\mathbf{W}), \tag{2.4}$$

where $\mathbf{Y} = [\mathbf{Y}^{(1)}, \mathbf{Y}^{(2)}, \cdots, \mathbf{Y}^{(n)}] \in \mathbb{R}^{m \times n}$ $(m < d)$ is an embedding matrix in which $\mathbf{Y}^{(i)} \in \mathbb{R}^m$ for $i = 1, 2, \cdots, n$ denotes the representation of data point $\mathbf{X}^{(i)}$ in the obtained low-dimensional subspace. $\mathcal{L}(\cdot)$ denotes a loss function, and $\mathcal{R}(\cdot)$ denotes a regularization function on the transformation matrix $\mathbf{W} \in \mathbb{R}^{m \times d}$. The methods differ in their choice of embedding $\mathbf{Y}$ and loss and regularization functions; some examples are presented below.

Multi-cluster feature selection (MCFS) [18] and minimum redundancy spectral feature selection (MRSF) [19] are two long-standing and well-known subspace learning-based unsupervised feature selection methods. In MCFS, the embedding $\mathbf{Y} \in \mathbb{R}^{m \times n}$

of each data $\mathbf{X}$ is first learned based on spectral clustering. After that, all data points are regressed to the learned embedding through a transformation matrix $\mathbf{W} \in \mathbb{R}^{m\times d} = [\mathbf{W}^{(1)}; \mathbf{W}^{(2)}; \cdots ; \mathbf{W}^{(m)}]$. The loss function is set to the Frobenius norm of the linear transformation error and the regularization function is set to the $\ell_{1,1}$ norm of the transformation matrix, which promotes sparsity. Thus, MCFS can be formulated mathematically as the set of separate optimization problems

$$\min_{\mathbf{W}^{(q)}} \|\mathbf{Y}^{(q)} - \mathbf{W}^{(q)}\mathbf{X}\|_2^2 + \lambda\|\mathbf{W}^{(q)}\|_1, \tag{2.5}$$

where $\mathbf{W}^{(q)} \in \mathbb{R}^d$ and $\mathbf{Y}^{(q)} \in \mathbb{R}^n$ are the qth rows of $\mathbf{W}$ and $\mathbf{Y}$, respectively, for $q = 1, 2, \cdots , m$. A score for each feature is measured by the maximum absolute value of the corresponding row of the transformation matrix:

$$MCFS(q) = \max_{p=1,2,\cdots,d} |\mathbf{W}^{(p,q)}| = \|W^{(q)}\|_\infty, \tag{2.6}$$

This score is then used in a filter-based feature selection scheme. MRSF is an extension of MCFS that changes the regularization function to an $\ell_{2,1}$-norm that enforce column sparsity on the transformation matrix. Ideally, the selected features should be representative enough to keep the loss value close to that obtained when using all features. In order to achieve feature selection, we expect that $\mathbf{W}$ holds a sparsity property with its columns, which means only a subset of the columns are nonzeros. We use the ℓ_2-norm of a $\mathbf{W}$ column to measure the importance of the corresponding feature, leading to an $\ell_{2,1}$-norm regularization function. Furthermore, MRSF ranks the importance of each feature according to the ℓ_2-norm of the corresponding column of the transformation matrix. Both MCFS and MRSF are able to select features that best preserve the subspace structure of the data due to the application of spectral clustering. However, the performance of these two methods is often degraded by the separate nature of subspace learning and feature selection [29]. In order to address this problem, many approaches on joint subspace learning and feature selection have been proposed. For example, in Gu et. al. [20] data are linearly projected to a low-dimensional subspace with a transformation matrix, and the local data geometric structure captured by a nearest neighbor graph is preserved in data embeddings on low-dimensional subspace. In the meanwhile, an $\ell_{2,1}-$norm is performed on transformation matrix to guide feature selection simultaneously. That is, subspace learning and feature selection are not two separate steps but combined into a single framework. Studies like [21–25] made further modifications to [20]: besides combining subspace learning and feature selection into a single framework, these methods also exploit the discriminative information of the data for unsupervised feature selection. For example, in unsupervised discriminative feature selection (UDFS) [21], data instances are assumed to come from c classes. UDFS uses local data geometric structure, which is based on the k-nearest neighbor set of each data point, to incorporate local data discriminative information into a feature selection framework. Like MCFS and MRFS, UDFS also assumes the existence of a transformation matrix $\mathbf{W} \in \mathbb{R}^{m\times c}$ that maps data to a low-dimensional space. The objective function of UDFS is

$$\min_{\mathbf{W}^T\mathbf{W}} \mathrm{Tr}(\mathbf{W}^T\mathbf{M}\mathbf{W}) + \alpha\|\mathbf{W}\|_{2,1}, \tag{2.7}$$

where $\mathbf{M}$ is an elaborate matrix that contains local data discriminative information; see [21] for details. One drawback of these discriminative exploitation feature selection methods is that the feature selection performance relies on an accurate estimation of number of classes.

Instead of projecting data onto a low-dimensional subspace, some approaches consider combining unsupervised feature selection methods with self-representation. In these methods, each feature is assumed to be representable as a linear combination of all (other) features, i.e., $\mathbf{X} = \mathbf{W}\mathbf{X} + \mathbf{E}$, where $\mathbf{W} \in \mathbb{R}^{d\times d}$ is a representation matrix and $\mathbf{E} \in \mathbb{R}^{d\times n}$ denotes a reconstruction error. That is, the data are linearly projected into the same data space so that the relationships between features can be gleaned from the transformation matrix. This type of method can be regarded as a special case of subspace learning-based feature selection methods where the embedding subspace is equal to the original space. Zhu et. al. [26] proposed a regularized self-representation (RSR) model for unsupervised feature selection that sets both the loss function and the regularization function to $\ell_{2,1}$-norms on the representation error $\mathbf{E}$ (for robustness to outlier samples) and transformation matrix $\mathbf{W}$ (for feature selection), respectively. RSR can therefore be written as

$$\min_{\mathbf{W}} \|\mathbf{X} - \mathbf{W}\mathbf{X}\|_{2,1} + \lambda\|\mathbf{W}\|_{2,1}. \tag{2.8}$$

RSR has been extended to non-convex RSR [27], where the regularization function is instead set to an $\ell_{2,p}$-norm for $0 < p < 1$. Unsupervised graph self-representation sparse feature selection (GSR_SFS) [28] further extends [27] by changing the loss function to a Frobenius norm, as well as by considering local data geometric structure preservation on embedding $\mathbf{W}\mathbf{X}$ through spectral graph analysis. GSR_SFS can be written in the following formulation

$$\min_{\mathbf{W}} \frac{1}{2}||\mathbf{X} - \mathbf{W}\mathbf{X}||_F^2 + \lambda_1 \mathrm{Tr}(\mathbf{X}^{\mathrm{T}}\mathbf{W}^{\mathrm{T}}\mathbf{L}\mathbf{W}\mathbf{X}) + \lambda_2||\mathbf{W}||_{2,1}, \tag{2.9}$$

where $\mathbf{L}$ is the graph Laplacian matrix. $\mathbf{L}$ can be calculated through $\mathbf{L} = \mathbf{D} - \mathbf{A}$, where $\mathbf{A}$ is the adjacency matrix which stores the similarities between vertices, and $\mathbf{D}$ is the degree matrix which is a diagonal matrix containing information about the degree of vertices. Self-representation based dual-graph regularized feature selection clustering (DFSC) [29] considers the error of self-representation for both the columns and the rows of $\mathbf{X}$ (i.e., both for features and data samples). Moreover, spectral graph analysis on both domains is considered. Subspace clustering guided unsupervised feature selection (SCUFS) [30] combines both self-representation and subspace clustering with unsupervised feature selection. In addition, SCUFS also exploits discriminative information for feature selection.

2.5 Self-Taught Learning

Transfer learning methods can be classified as homogeneous and heterogeneous. Homogeneous transfer learning methods assume $\mathcal{X}_{\text{src}} = \mathcal{X}_{\text{trg}}$ while heterogeneous transfer learning methods assume $\mathcal{X}_{\text{src}} \neq \mathcal{X}_{\text{trg}}$. We focus on homogeneous transfer learning.

Self-taught learning can be categorized into the group of inductive transfer learning methods [43], in which $\mathcal{T}_{\text{trg}} \neq \mathcal{T}_{\text{src}}$ while the domains can be either same or different. The idea of self-taught learning was first proposed by Raina et. al. [45] and implemented through dictionary learning and sparse coding.[2] To be more specific, a dictionary is learned using source samples:

$$\begin{aligned}&\min_{\mathbf{D},\mathbf{A}_{\text{src}}} \|\mathbf{X}_{\text{src}} - \mathbf{D}\mathbf{A}_{\text{src}}\|_F^2 + \beta \sum_{i=1}^{n_{\text{src}}} \|\mathbf{A}_{\text{src}}^{(i)}\|_1,\\ &\text{s.t. } \|\mathbf{D}^{(j)}\| \leq 1, 1 \leq j \leq s,\end{aligned} \tag{2.10}$$

where $\mathbf{D} \in \mathbb{R}^{d\times s}$ is a dictionary with each column as a dictionary element, and where each column in $\mathbf{A}_{\text{src}} \in \mathbb{R}^{s\times n_{\text{src}}}$ represents the sparse coefficient vector of the corresponding unlabeled source sample from $\mathbf{X}_{\text{src}} \in \mathbb{R}^{d\times n_{\text{src}}}$. After the dictionary $\mathbf{D}$ is obtained, a new labeled training set $\{\mathbf{A}_{\text{trg}}, \mathbf{Y}_{\text{trg}}\}$ in the target domain is computed through

$$\min_{\mathbf{A}_{\text{trg}}} \|\mathbf{X}_{\text{trg}} - \mathbf{D}\mathbf{A}_{\text{trg}}\|_F^2 + \beta \sum_{i=1}^{n_{\text{trg}}} \|\mathbf{A}_{\text{trg}}^{(i)}\|_1, \tag{2.11}$$

where each column in $\mathbf{A}_{\text{trg}} \in \mathbb{R}^{s\times n_{\text{trg}}}$ represents the sparse coefficient vector of the corresponding labeled target sample from $\mathbf{X}_{\text{trg}} \in \mathbb{R}^{d\times n_{\text{trg}}}$. Finally, a classifier is learned

[2]We use abbreviation STL to denote the method of [45] in the sequel, while we use the full name “self-taught learning“ for the class of learning problems.

on the new labeled training set by applying a supervised learning algorithm. The idea of self-taught learning has been applied in scenarios such as clustering [46], visual tracking [47], object localization [48, 49], hyperspectral image classification [50], wound infection detection [51], etc.

Wang et. al. [52] propose robust and discriminative self-taught learning (RDSTL) as an extension to STL. Compared with STL, two changes are made in order to increase the robustness of the learning model and make use of supervision information contained in target samples. The first is to replace the ℓ_1-norm loss function used in STL with an $\ell_{2,1}$-norm loss function because the latter is claimed to be more robust to noise and outliers. The second is to take advantage of label information of target samples during learning. Assume $\mathbf{X}_k \in \mathbb{R}^{d\times n_k}$ and $\mathbf{A}_k \in \mathbb{R}^{s\times n_k}$ denote the samples and corresponding sparse codes belonging to the kth class. We refer to source samples as belonging to the 0^{th} class and assume that the dataset $\mathbf{X}$ is arranged by classes so that $\mathbf{X} = [\mathbf{X}_0, \mathbf{X}_1, \cdots, \mathbf{X}_K]$, where K is the total number of classes in the target samples, with $\mathbf{A}$ following the same setup. Then RDSTL can be written as the following optimization problem:

$$\begin{aligned} &\min_{\mathbf{D},\mathbf{A}} \|\mathbf{X} - \mathbf{D}\mathbf{A}\|_F^2 + \beta \sum_{k=0}^{K} \|\mathbf{A}_k^{\mathrm{T}}\|_{2,1}, \\ &\text{s.t. } \|\mathbf{D}^{(j)}\| \le 1, 1 \le j \le s. \end{aligned} \tag{2.12}$$

Another advantage of imposing $\ell_{2,1}$-norm regularization on the representation coefficients is that it makes the learning process insensitive to the dictionary size. This is because sparsity on rows of $\mathbf{A}$ helps select basis vectors in $\mathbf{D}$: a basis vector contributes little to data representation if the ℓ_2-norm value of its corresponding coefficient vector is close to 0. Therefore, the final task performance should not be sensitive to the dictionary size once it is large enough.

Li et. al. [53] proposes a self-taught low-rank (S-Low) coding framework which is suitable for both clustering and classification tasks in visual learning. By imposing a low-rank constraint onto the sparse coefficient matrix, S-Low coding is claimed to be able to characterize the global structure information in the target domain. The objective function of S-Low coding is

$$\begin{aligned}\min_{\substack{\mathbf{D},\mathbf{A}_{\text{src}},\mathbf{A}_{\text{trg}},\\ \mathbf{E}_{\text{src}},\mathbf{E}_{\text{trg}}}} \; & \|\mathbf{A}_{\text{trg}}\|_{\gamma_1} + \lambda_1 M_{\gamma_2}(\mathbf{E}_{\text{src}}) + \lambda_2 M_{\gamma_2}(\mathbf{E}_{\text{trg}}) + \lambda_3 \|\mathbf{A}_{\text{src}}\|_{2,1}, \\ \text{s.t. } & \mathbf{X}_{\text{src}} = \mathbf{D}\mathbf{A}_{\text{src}} + \mathbf{E}_{\text{src}},\; \mathbf{X}_{\text{trg}} = \mathbf{D}\mathbf{A}_{\text{trg}} + \mathbf{E}_{\text{trg}},\end{aligned} \tag{2.13}$$

where $\|\cdot\|_{\gamma_1}$ denotes the matrix γ-norm with parameter γ_1 and $M_{\gamma_2}(\cdot)$ denotes the minimax concave penalty norm with parameter γ_2. We refer readers to [53] for more details on the roles and definitions of these two norms.

Though the self-taught learning approaches mentioned above use different schemes for knowledge transfer, they all use the whole source sample set without considering their relevance to target domain, which makes these methods potentially vulnerable to negative transfer.

2.6 Human Activity Recognition

Human activity recognition (HAR) is a technique that aims at learning high-level knowledge about human activities from the low-level sensor inputs [62]. In recent years, the growing ubiquity of sensor-equipped wearables such as smart wristbands and smartphones have significantly promoted researches regarding HAR in the field of pervasive computing [63].

Machine learning algorithms have been widely used in HAR. In the last decade, traditional machine learning tools such as Markov models [64, 65] and decision trees [66, 67] have yielded tremendous progress in HAR. However, traditional machine learning methods suffer from several limitations in HAR using wearables. Most traditional machine learning algorithms applied in HAR use manually designed features including mean, variance, and frequency, which are shallow and heavily rely on human domain knowledge and experience; furthermore, they are specific to particular tasks. The drawback of these hand-crafted features is two-fold: 1. shallow features can only be used to recognize low-level activities like sitting and standing but are hardly capable for high-level activities like printing papers and attending a seminar [68]; 2. task-specific features can hardly be transferred to different environments or tasks. Therefore, traditional machine learning algorithms cannot handle complex HAR scenarios, and they require one specifically designed model for each task, which increases the time and labor cost to build HAR systems in terms of both labeled data collection and model construction.

In recent years, the application of deep learning methods to HAR has significantly alleviated the drawbacks of traditional machine learning based HAR methods. First, deep neural network can extract high-level features with little or no human design. Second, deep learning models can be reused for similar tasks, which makes HAR model construction more efficient. Different deep learning models such as deep neural networks [69, 70], convolutional neural networks [71, 72], autoencoders [73, 74], restricted Boltzmann machines [75, 76], and recurrent neural networks [77, 78] have been applied in HAR. We refer readers to [62] for more details on deep learning based HAR.

2.7 Few-Shot Learning

Few-shot learning (FSL) is a transfer learning technique that applies knowledge from existing data to data from unseen classes which do not have sufficient labeled training data for model training. For example, if we are performing a task of recognizing bird species from images, the number of data for some rare species of birds may be insufficient to be used for training. Then the knowledge of existing bird species can be borrowed for the learning of new species. In this case, the machine learning problem with a classifier for bird images with insufficient amount of training data is treated as a FSL problem. If there is only one image of a bird species, this would be a one-shot learning problem.

The first work for FSL is [79], in which a variational Bayesian framework is proposed to represent visual object categories as probabilistic models. Existing object categories, denoted as prior knowledge, is represented as a probability density function on the parameters of these models. While unseen categories, denoted as the posterior model, is obtained by updating the prior with one or more observations. Lim et al. [80] propose a sample-borrowing method for multiclass object detection that adds selected samples from similar categories to the training set in order to increase the number of training data.

In recent years, deep learning based FSL has become the mainstream of FSL due to their unparalleled performance. Initialization based methods [54, 81, 82] focus on the fine-tuning process for new tasks. Finn et al. [81] aims to learn a good initialization for new tasks. Ravi et al. [54] and Munkhdalai [82] replace the weight-update process with an external memory. Hallucination based methods [83–85] learn a data generator from the base classes and use the learned generator to hallucinate data for new classes. Hariharan et al. [83] transfers variance from base class data to new classes. Antoniou et al. [84] uses generative adversarial networks to transfer style from base classes to new classes. Wang et al. [85] integrate the generator into a meta-learning framework for transfer learning purposes. Distance metric learning based methods [55, 56, 86, 87] measure the distance between two images and take advantage of the distance to classify unseen images. Examples of distance metrics include cosine similarity [56], Euclidean distance [55], CNN-based relation module [86], and graph neural network [87].

CHAPTER 3
UNSUPERVISED FEATURE SELECTION

3.1 Introduction

Feature selection is a dimensionality reduction technique that selects a subset of representative features from highdimensional data by eliminating irrelevant and redundant features. Recently, feature selection combined with sparse learning has attracted significant attention due to its outstanding performance compared with traditional feature selection methods that ignores correlation between features. These works first map data onto a low-dimensional subspace and then select features by posing a sparsity constraint on the transformation matrix. However, they are restricted by design to linear data transformation, a potential drawback given that the underlying correlation structures of data are often non-linear. To leverage a more sophisticated embedding, we propose an autoencoder-based unsupervised feature selection approach that leverages a single-layer autoencoder for a joint framework of feature selection and manifold learning. More specifically, we enforce column sparsity on the weight matrix connecting the input layer and the hidden layer, as in previous work. Additionally, we include spectral graph analysis on the projected data into the learning process to achieve local data geometry preservation from the original data space to the low-dimensional feature space. Extensive experiments are conducted on image, audio, text, and biological data. The promising experimental results validate the superiority of the proposed method[1].

[1]Contents presented in this chapter have been published as a journal paper [88].

3.2 Proposed Method

In this section, we introduce our proposed graph autoencoder-based unsupervised feature selection (GAFS). Our proposed framework performs broad data structure preservation through a single-layer autoencoder and also preserves local data geometric structure through spectral graph analysis. In contrast to existing methods that exploit discriminative information for unsupervised feature selection by imposing orthogonal constraints on the transformation matrix [21] or low-dimensional data representation [22, 23], GAFS does not include such constraints. More specifically, we do not add orthogonal constraints on the transformation matrix because feature weight vectors are not necessarily orthogonal with each other in real-world applications [89], allowing GAFS to be applicable to a larger set of applications [17]. Furthermore, methods posing orthogonal constraints on low-dimensional data representations makes a good estimation of number of classes necessary to obtain reliable label indicators for those algorithms; such estimation is difficult to achieve in an unsupervised framework.

3.2.1 Objective Function

The objective function of GAFS includes three parts: a term based on a single-layer autoencoder promoting broad data structure preservation; a term based on spectral graph analysis promoting local data geometric structure preservation; and a regularization term promoting feature selection. As mentioned in Chapter 2.2, a single-layer autoencoder aims at minimizing the reconstruction error between output and input data by optimizing a reconstruction error-driven loss function:

$$\mathcal{L}(\mathbf{\Theta}) = \frac{1}{2n}\sum_{i=1}^{n} \|\mathbf{X}^{(i)} - h(\mathbf{X}^{(i)};\mathbf{\Theta})\|_2^2 = \frac{1}{2n}\|\mathbf{X} - h(\mathbf{X};\mathbf{\Theta})\|_F^2, \tag{3.1}$$

where $\mathbf{\Theta} = [\mathbf{W}_1, \mathbf{W}_2, \mathbf{b}_1, \mathbf{b}_2]$, $h(\mathbf{X};\mathbf{\Theta}) = \sigma\left(\mathbf{W}_2 \cdot \sigma(\mathbf{W_1}\mathbf{X} + \mathbf{b}_1) + \mathbf{b}_2\right)$; we use the sigmoid function as the activation function: $\sigma(z) = 1/(1 + \exp(-z))$.

Since $\mathbf{W}_1$ is a weight matrix applied directly on the input data, each column of $\mathbf{W}_1$ can be used to measure the importance of the corresponding data feature. Therefore, $\mathcal{R}(\Theta) = \|\mathbf{W}_1\|_{2,1}$ can be used as a regularization function to promote feature selection as detailed in Chapter 2.4. The objective function for the single-layer autoencoder based unsupervised feature selection can be obtained by combining this regularization function with the loss function of (4.2), providing us with the optimization

$$\min_{\boldsymbol{\Theta}} \frac{1}{2n}\|\mathbf{X} - h(\mathbf{X}; \boldsymbol{\Theta})\|_F^2 + \lambda\|\mathbf{W}_1\|_{2,1}, \tag{3.2}$$

where λ is a balance parameter.

Local geometric structures of the data often contain discriminative information of neighboring data point pairs [18]. They assume that nearby data points should have similar representations. It is often more efficient to combine both broad and local data information during low-dimensional subspace learning [90]. In order to characterize the local data geometric structure, we construct a k-nearest neighbor (kNN) graph $\mathbb{G}$ on the data space. The edge weight between two connected data points is determined by the similarity between those two points. We choose cosine distance as similarity measurement for its simplicity. Therefore the adjacency matrix $\mathbf{A}$ for the graph $\mathbb{G}$ is defined as

$$\mathbf{A}^{(i,j)} = \begin{cases} \dfrac{\mathbf{X}^{(i)^T}\mathbf{X}^{(j)}}{\|\mathbf{X}^{(i)}\|_2\|\mathbf{X}^{(j)}\|_2} & \text{if } \mathbf{X}^{(i)} \in \mathcal{N}_k(\mathbf{X}^{(j)}) \text{ or } \mathbf{X}^{(j)} \in \mathcal{N}_k(\mathbf{X}^{(i)}), \\ 0 & \text{otherwise,} \end{cases} \tag{3.3}$$

where $\mathcal{N}_k(\mathbf{X}^{(i)})$ denotes the k-nearest neighborhood set for $\mathbf{X}^{(i)}$, and $\mathbf{X}^{(i)^T}$ refers to the transpose of $\mathbf{X}^{(i)}$. The Laplacian matrix $\mathbf{L}$ of the graph $\mathbb{G}$ is defined as $\mathbf{L} = \mathbf{D} - \mathbf{A}$, where $\mathbf{D}$ is a diagonal matrix whose i^{th} element on the diagonal is defined as $\mathbf{D}^{(i,i)} = \sum_{j=1}^{n} \mathbf{A}^{(i,j)}$.

In order to preserve the local data geometric structure in the learned subspace (i.e., if two data points $\mathbf{X}^{(i)}$ and $\mathbf{X}^{(j)}$ are close in original data space then the corresponding low-dimensional representations $\mathbf{Y}^{(i)}$ and $\mathbf{Y}^{(j)}$ are also close in the low-dimensional embedding space), we set up the following minimization objective:

$$\begin{aligned}\mathcal{G}(\boldsymbol{\Theta}) &= \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\|\mathbf{Y}^{(i)}-\mathbf{Y}^{(j)}\|_2^2\mathbf{A}^{(i,j)} \\ &= \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}(\mathbf{Y}^{(i)^T}\mathbf{Y}^{(i)}-\mathbf{Y}^{(i)^T}\mathbf{Y}^{(j)}-\mathbf{Y}^{(j)^T}\mathbf{Y}^{(i)}+\mathbf{Y}^{(j)^T}\mathbf{Y}^{(j)})\mathbf{A}^{(i,j)} \\ &= \sum_{i=1}^{n}\mathbf{Y}^{(i)^T}\mathbf{Y}^{(i)}\mathbf{D}^{(i,i)}-\sum_{i=1}^{n}\sum_{j=1}^{n}\mathbf{Y}^{(i)^T}\mathbf{Y}^{(j)}\mathbf{A}^{(i,j)} \\ &= \mathrm{Tr}(\mathbf{Y}(\boldsymbol{\Theta})\mathbf{D}\mathbf{Y}(\boldsymbol{\Theta})^{\mathrm{T}})-\mathrm{Tr}(\mathbf{Y}(\boldsymbol{\Theta})\mathbf{A}\mathbf{Y}(\boldsymbol{\Theta})^{\mathrm{T}})=\mathrm{Tr}(\mathbf{Y}(\boldsymbol{\Theta})\mathbf{L}\mathbf{Y}(\boldsymbol{\Theta})^{\mathrm{T}}),\end{aligned} \tag{3.4}$$

where $\mathrm{Tr}(\cdot)$ denotes the trace operator, $\mathbf{Y}^{(i)}(\boldsymbol{\Theta})=\sigma(\mathbf{W_1}\mathbf{x}^{(i)}+\mathbf{b}_1)$ for $i=1,2,\cdots,n$ (and we often drop the dependence on $\boldsymbol{\Theta}$ for readability), and $\mathbf{Y}(\boldsymbol{\Theta})=[\mathbf{Y}^{(1)}(\boldsymbol{\Theta}),\mathbf{Y}^{(2)}(\boldsymbol{\Theta}),\cdots,\mathbf{Y}^{(n)}($

Therefore, by combining the single-layer autoencoder based feature selection objective (3.2) and the local data geometric structure preservation into consideration, the resulting objective function of GAFS can be written in terms of the following minimization with respect to the parameters $\boldsymbol{\Theta}=[\mathbf{W}_1,\mathbf{W}_2,\mathbf{b}_1,\mathbf{b}_2]$:

$$\begin{aligned}\hat{\boldsymbol{\Theta}} &= \arg\min_{\boldsymbol{\Theta}}\mathcal{F}(\boldsymbol{\Theta})=\arg\min_{\boldsymbol{\Theta}}\mathcal{L}(\boldsymbol{\Theta})+\mathcal{R}(\boldsymbol{\Theta})+\mathcal{G}(\boldsymbol{\Theta}) \\ &= \arg\min_{\boldsymbol{\Theta}}\left[\frac{1}{2n}\|\mathbf{X}-h(\mathbf{X};\boldsymbol{\Theta})\|_F^2+\lambda\|\mathbf{W}_1\|_{2,1}+\gamma\mathrm{Tr}(\mathbf{Y}\mathbf{L}\mathbf{Y}^{\mathrm{T}})\right],\end{aligned} \tag{3.5}$$

where λ and γ are two balance parameters. Filter-based feature selection is then performed using the score function $GAFS(q)=\|\mathbf{W}_1^{(q)}\|_2$ based on the weight matrix $\mathbf{W}_1$ from $\hat{\boldsymbol{\Theta}}$.

3.2.2 Optimization

The objective function of GAFS shown in (4.6) does not have a closed-form solution. In this work, we implement the L-BFGS algorithm [91] using the *minFunc* toolbox [92] to solve the GAFS optimization problem. The solver requires the gradients of the objective function in (4.6) with respect to its parameters $\mathbf{\Theta}$.

The gradients for the loss term $\mathcal{L}(\mathbf{\Theta})$ can be obtained through a back-propagation algorithm. We defer the details for the derivation of the gradients of the error term, which are standard in the formulation of backpropagation for an autoencoder. The resulting gradients are as follows:

$$\begin{aligned} \frac{\partial \mathcal{L}(\mathbf{\Theta})}{\partial \mathbf{W}_1} &= \frac{1}{n}\mathbf{\Delta_2}\mathbf{X}^T, \\ \frac{\partial \mathcal{L}(\mathbf{\Theta})}{\partial \mathbf{W}_2} &= \frac{1}{n}\mathbf{\Delta_3}\mathbf{Y}^T, \\ \frac{\partial \mathcal{L}(\mathbf{\Theta})}{\partial \mathbf{b}_1} &= \frac{1}{n}\sum_{i=1}^{n}\mathbf{\Delta_2}^{(i)} = \frac{1}{n}\mathbf{\Delta_2}\mathbf{1}, \\ \frac{\partial \mathcal{L}(\mathbf{\Theta})}{\partial \mathbf{b}_2} &= \frac{1}{n}\sum_{i=1}^{n}\mathbf{\Delta_3}^{(i)} = \frac{1}{n}\mathbf{\Delta_3}\mathbf{1}. \end{aligned} \tag{3.6}$$

Each column $\mathbf{\Delta}_2^{(i)}$ and $\mathbf{\Delta}_3^{(i)}$ of $\mathbf{\Delta_2} \in \mathbb{R}^{m\times n}$ and $\mathbf{\Delta_3} \in \mathbb{R}^{d\times n}$, respectively, contains the error term of the corresponding data point for the hidden layer and the output layer, respectively, having entries as follows:

$$\begin{aligned} \mathbf{\Delta}_3^{(p,i)} &= (\bar{\mathbf{X}}^{(p,i)} - \mathbf{X}^{(p,i)}) \cdot \bar{\mathbf{X}}^{(p,i)} \cdot (1 - \bar{\mathbf{X}}^{(p,i)}), \\ \mathbf{\Delta}_2^{(q,i)} &= \left(\sum_{p=1}^{d} \mathbf{W}_2^{(q,p)} \mathbf{\Delta}_3^{(p,i)}\right) \cdot \mathbf{Y}^{(q,i)} \cdot (1 - \mathbf{Y}^{(q,i)}), \end{aligned} \tag{3.7}$$

for $p = 1, 2, \cdots, d$, $q = 1, 2, \cdots, m$, and $i = 1, 2, \cdots, n$, and where $\bar{\mathbf{X}}$ denotes the reconstructed data output of the autoencoder. Equation (3.7) can also be rewritten in matrix form as

$$\begin{aligned} \mathbf{\Delta_3} &= (\bar{\mathbf{X}} - \mathbf{X}) \bullet \bar{\mathbf{X}} \bullet (\mathbf{1} - \bar{\mathbf{X}}), \\ \mathbf{\Delta_2} &= (\mathbf{W}_2^T \mathbf{\Delta_3}) \bullet \mathbf{Y} \bullet (\mathbf{1} - \mathbf{Y}), \end{aligned} \tag{3.8}$$

where $\bullet$ denotes the element-wise product operator. In the sequel, we use $\mathbf{1}$ and $\mathbf{0}$ to denote an all-ones and all-zeros matrix or vector with of the appropriate size, respectively.

The regularization term $R(\mathbf{\Theta}) = \|\mathbf{W}_1\|_{2,1}$, whose derivative does not exist for its ith column $\mathbf{W}_1^{(i)}$ when $\mathbf{W}_1^{(i)} = \mathbf{0}$ for $i = 1, 2, \cdots, d$. In this case,

$$\frac{\partial \mathcal{R}(\mathbf{\Theta})}{\partial \mathbf{W}_1} = \mathbf{W_1 U}, \tag{3.9}$$

where $\mathbf{U} \in \mathbb{R}^{d \times d}$ is a diagonal matrix whose ith element on the diagonal is

$$\mathbf{U}^{(i,i)} = \begin{cases} \left(\|\mathbf{W}_1^{(i)}\|_2 + \epsilon\right)^{-1}, & \|\mathbf{W}_1^{(i)}\|_2 \neq 0, \\ 0, & \text{otherwise.} \end{cases} \tag{3.10}$$

where ϵ is a small constant added to avoid overflow [29]. Since $\|\mathbf{W}_1\|_{2,1}$ is not differentiable at $\mathbf{0}$, we calculate the subgradient for each element in $\mathbf{W}_1$ in that case. That is, for each element in $\mathbf{W}_1$, the subgradient at $\mathbf{0}$ can be an arbitrary value in the interval $[-1, 1]$, and so we set the gradient to 0 for computational convenience. In summary, the gradients for the regularization term is:

$$\begin{aligned} \frac{\partial \mathcal{R}(\mathbf{\Theta})}{\partial \mathbf{W}_1} &= \lambda \mathbf{W_1 U}, \\ \frac{\partial \mathcal{R}(\mathbf{\Theta})}{\partial \mathbf{W}_2} &= \mathbf{0}, \\ \frac{\partial \mathcal{R}(\mathbf{\Theta})}{\partial \mathbf{b}_1} &= \mathbf{0}, \\ \frac{\partial \mathcal{R}(\mathbf{\Theta})}{\partial \mathbf{b}_2} &= \mathbf{0}, \end{aligned} \tag{3.11}$$

The gradients of the graph term $\mathcal{G}(\mathbf{\Theta}) = \gamma \mathrm{Tr}(\mathbf{YLY}^{\mathrm{T}})$ can be obtained in a straightforward fashion as follows:

$$\begin{aligned} \frac{\partial \mathcal{L}(\mathbf{\Theta})}{\partial \mathbf{W}_1} &= \frac{\partial \mathrm{Tr}(\gamma \mathbf{YLY}^{\mathrm{T}})}{\partial \mathbf{Y}} \cdot \frac{\partial \mathbf{Y}}{\partial \mathbf{Z}} \cdot \frac{\partial \mathbf{Z}}{\partial \mathbf{W}_1} = 2\gamma \left(\mathbf{YL} \bullet \mathbf{Y} \bullet (\mathbf{1} - \mathbf{Y})\right) \mathbf{X}^T, \\ \frac{\partial \mathcal{L}(\mathbf{\Theta})}{\partial \mathbf{b}_1} &= \frac{\partial \mathrm{Tr}(\gamma \mathbf{YLY}^{\mathrm{T}})}{\partial \mathbf{Y}} \cdot \frac{\partial \mathbf{Y}}{\partial \mathbf{Z}} \cdot \frac{\partial \mathbf{Z}}{\partial \mathbf{b}_1} = 2\gamma \left(\mathbf{YL} \bullet \mathbf{Y} \bullet (\mathbf{1} - \mathbf{Y})\right) \mathbf{1}, \\ \frac{\partial \mathcal{L}(\mathbf{\Theta})}{\partial \mathbf{W}_2} &= \mathbf{0}, \\ \frac{\partial \mathcal{L}(\mathbf{\Theta})}{\partial \mathbf{b}_2} &= \mathbf{0}. \end{aligned} \tag{3.12}$$

To conclude, the gradients of the GAFS objective function with respect to $\mathbf{\Theta} = [\mathbf{W}_1, \mathbf{W}_2, \mathbf{b}_1, \mathbf{b}_2]$ can be written as

$$\begin{aligned}
\frac{\partial \mathcal{F}(\mathbf{\Theta})}{\partial \mathbf{W}_1} &= \frac{1}{n}\mathbf{\Delta_2}\mathbf{X}^T + \lambda \mathbf{W_1}\mathbf{U} + 2\gamma \left(\mathbf{YL} \bullet \mathbf{Y} \bullet (\mathbf{1} - \mathbf{Y})\right)\mathbf{X}^T, \\
\frac{\partial \mathcal{F}(\mathbf{\Theta})}{\partial \mathbf{W}_2} &= \frac{1}{n}\mathbf{\Delta_3}\mathbf{Y}^T, \\
\frac{\partial \mathcal{F}(\mathbf{\Theta})}{\partial \mathbf{b}_1} &= \frac{1}{n}\mathbf{\Delta_2}\mathbf{1} + 2\gamma \left(\mathbf{YL} \bullet \mathbf{Y} \bullet (\mathbf{1} - \mathbf{Y})\right)\mathbf{1}, \\
\frac{\partial \mathcal{F}(\mathbf{\Theta})}{\partial \mathbf{b}_2} &= \frac{1}{n}\mathbf{\Delta_3}\mathbf{1}
\end{aligned} \tag{3.13}$$

3.3 Experiments

In this section, we evaluate the feature selection performance of GAFS in terms of both supervised and unsupervised tasks, e.g. clustering and classification, on several benchmark datasets. We also compare GAFS with other state-of-the-art unsupervised feature selection algorithms. To be more specific, we first select p representative features and then perform both clustering and classification on those selected features. The performance of clustering and classification is used as the metric to evaluate feature selection algorithms. We perform experiments on eight benchmark datasets,[2] including three image datasets (MNIST, COIL20, Yale), three text datasets (PCMAC, BASEHOCK, RELATHE), one audio dataset (Isolet), and one biological dataset (Prostate_GE). Detailed properties of those datasets are summarized in Table 4.1.

3.3.1 Evaluation Metric

We perform both supervised (i.e., classification) and unsupervised (i.e., clustering) tasks on datasets formulated by the selected features in order to evaluate the effectiveness of feature selection algorithms. For classification, we employ softmax classifier for its simplicity and compute the classification accuracy as the evaluation

[2] All datasets are downloaded from http://featureselection.asu.edu/datasets.php

Dataset	Features	Instances	Classes	Type
MNIST	784	60000	10	Image
COIL20	1024	1440	20	Image
Yale	1024	165	15	Image
PCMAC	3289	1943	2	Text
BASEHOCK	4862	1993	2	Text
RELATHE	4322	1427	2	Text
Prostate_GE	5966	102	2	Biology
Isolet	617	1560	26	Audio

Table 3.1. Details of datasets used.

metric for feature selection effectiveness. For clustering, we use k-means clustering on the selected features and use two different evaluation metrics to evaluate the clustering performance of all methods. The first is clustering accuracy (ACC), defined as

$$\mathrm{ACC} = \frac{1}{n}\sum_{i=1}^{n} \delta(g_i, \mathrm{map}(c_i)),$$

where n is the total number of data samples, $\delta(\cdot)$ is defined by $\delta(a, b) = 1$ when $a = b$ and 0 when $a \neq b$, $\mathrm{map}(\cdot)$ is the optimal mapping function between cluster labels and class labels obtained using the Hungarian algorithm [93], and c_i and g_i are the clustering and ground truth labels of a given data sample $\mathbf{x_i}$, respectively. The second is normalized mutual information (NMI), which is defined as

$$\mathrm{NMI} = \frac{\mathrm{MI}(C, G)}{\max(H(C), H(G))},$$

where C and G are clustering labels and ground truth labels, respectively, $\mathrm{MI}(C, G)$ is the mutual information between C and G, and $H(C)$ and $H(G)$ denote the entropy of C and G, respectively. More details about NMI are available in [94]. For both ACC and NMI, 20 clustering processes are repeated with random initialization for each case following the setup of [18] and [21], and we report the corresponding mean values of ACC and NMI.

3.3.2 Experimental Setup

In our last experiment, we compare GAFS with LapScore[3] [14], SPEC[4] [15], MRSF[5] [19], UDFS[6] [21], and RSR[7] [26]. Among these methods, LapScore and SPEC are filter feature selection methods which are based on data similarity. LapScore uses spectral graph analysis to set a score for each feature. SPEC is an extension to LapScore and can be applied to both supervised and unsupervised scenarios in which schemes of constructing graphs used for data similarity measurement are different. Details on MRSF, UDFS, and RSR can be found in Chapter 2.4. Besides the five methods, we also compare GAFS with the performance of using all features as the baseline.

Both GAFS and compared algorithms include parameters to adjust. In this experiment, we fix some parameters and tune others according to a "grid-search" strategy. For all algorithms, we select $p \in \{2\%, 4\%, 6\%, 8\%, 10\%, 20\%, 30\%, 40\%, 50\%, 60\%, 70\%, 80\%\}$ of all features for each dataset. For all graph-based algorithms, the number of nearest neighbor in a kNN graph is set to 5. For all algorithms projecting data onto a low-dimensional space, the space dimensionality is set in the range of $m \in \{10, 20, 30, 40\}$. In GAFS, the range for the hidden layer size is set to match that of the subspace dimensionality m,[8] while the balance parameters are given ranges $\lambda \in \{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1\}$ and $\gamma \in \{0, 10^{-4}, 5 \times 10^{-4}, 10^{-3}, 5 \times 10^{-3}\}$, respectively. For UDFS, we use the range $\gamma \in \{10^{-9}, 10^{-6}, 10^{-3}, 1, 10^{3}, 10^{6}, 10^{9}\}$, and λ

[3]Available at http://www.cad.zju.edu.cn/home/dengcai/Data/code/LaplacianScore.m

[4]Available at https://github.com/matrixlover/LSLS/blob/master/fsSpectrum.m

[5]Available at https://sites.google.com/site/alanzhao/Home

[6]Available at http://www.cs.cmu.edu/ yiyang/UDFS.rar

[7]Available at https://github.com/guangmingboy/githubs_doc

[8]We will alternatively use the terminologies *subspace dimensionality* and *hidden layer size* in descriptions of GAFS.

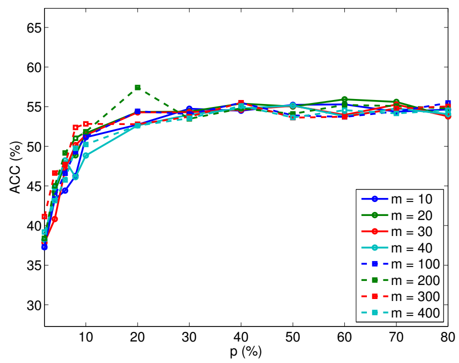
is fixed to 10^3. For RSR, we use the range $\lambda \in \{10^{-3}, 5\times10^{-3}, 10^{-2}, 5\times10^{-2}, 10^{-1}, 5\times 10^{-1}, 1, 5, 10, 10^2\}$.

For each specific value of p on a certain dataset, we tune the parameters for each algorithm in order to achieve the best results among all possible combinations. For classification, we report the highest classification accuracy. For clustering, we report the highest average values for both ACC and NMI from 20 repetitions.

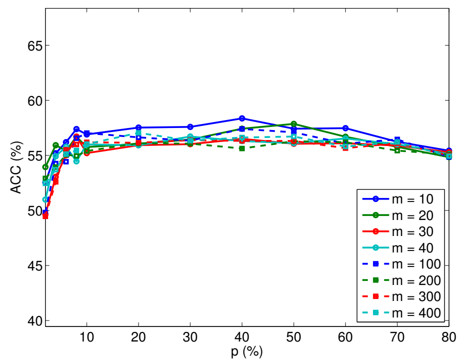
3.3.3 Parameter Sensitivity

We study the performance variation of GAFS with respect to the hidden layer size m and the two balance parameters λ and γ. We show the results on all the 8 datasets in terms of ACC.

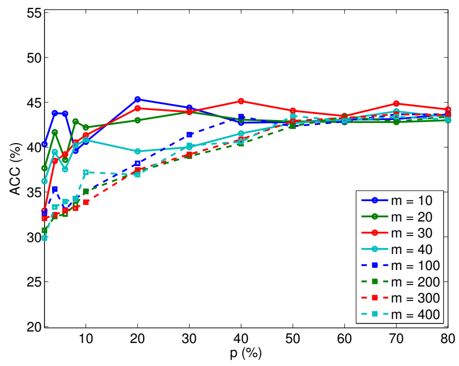
We first study the parameter sensitivity of GAFS with respect to subspace dimensionality m. Besides the aforementioned manifold dimensionality range $m \in \{10, 20, 30, 40\}$, which are common for both proposed and comparing algorithms, we also conducted experiments with hidden layer size values of $m \in \{100, 200, 300, 400\}$ to investigate the performance change for a larger range of reduced dimensionality values. The results in Fig. 4.1 show that the performance of GAFS is not too sensitive to hidden layer size on the given datasets, with the exception of Yale, where the performance with hidden layer size of $m \in \{10, 20, 30, 40\}$ is apparently better than that with reduced dimensionality $m \in \{100, 200, 300, 400\}$, while the performance variations are small in the latter set. One possible reason behind this behavior is that for a human face image dataset like Yale, the differences between data instances can be subtle since they may only lie in a small area of relevance such as eyes, mouth, nose, etc. Therefore, in this case a small subspace dimensionality can be enough for information preservation, while a large subspace dimensionality may introduce redundant information that may harm feature selection performance.



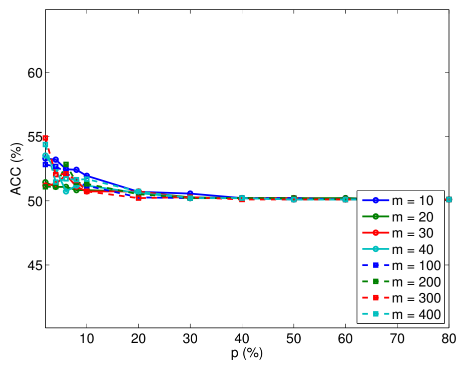
(a) MNIST



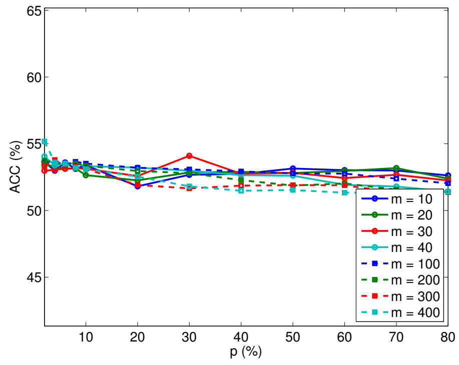
(b) COIL20



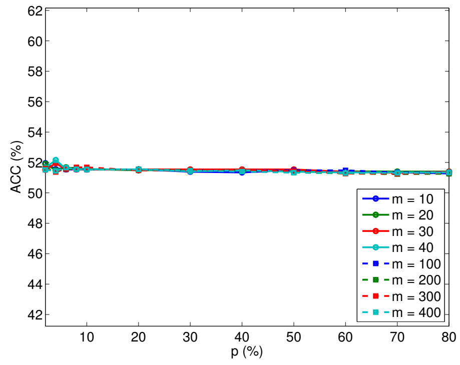
(c) Yale



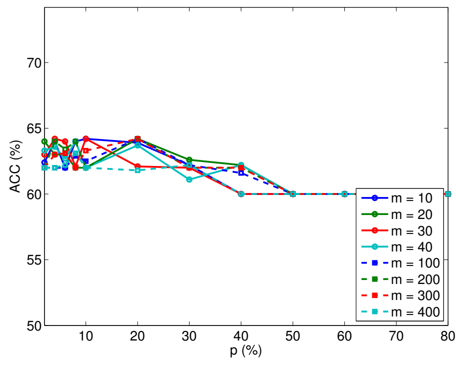
(d) PCMAC



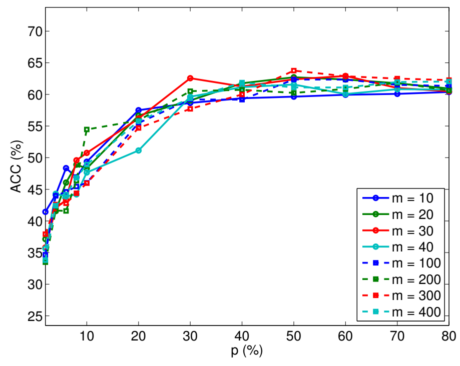
(e) BASEHOCK



(f) RELATHE



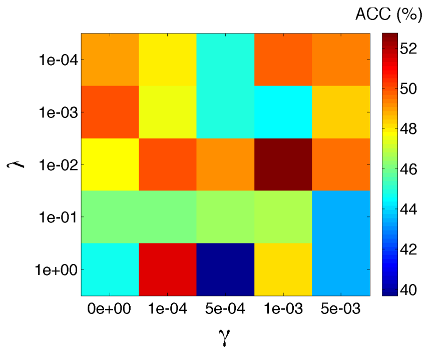
(g) Prostate_GE



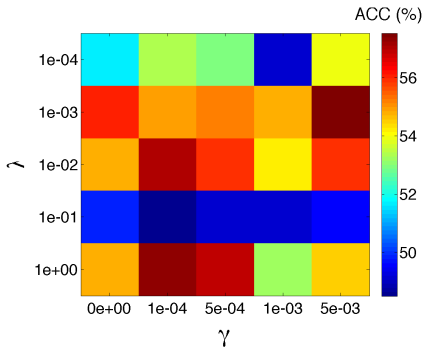
(h) Isolet

Figure 3.1. Performance variation of the GAFS w.r.t. dimensionality of subspace m and the percentage of features selected p (%).

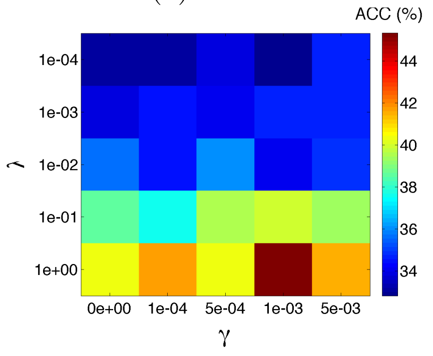
We also study the performance of GAFS on balance parameters λ and γ, with fixed percentage of selected features and hidden layer size. We set $p = 20\%$, as Fig. 4.1 shows that the performance stabilizes starting at that value of p. For subspace dimensionality, we choose $m = 10$ since Fig. 4.1 shows that the performance of GAFS is not sensitive to the value of m. The performance results are shown in Fig. 3.2, where we find that different datasets present different trends on the ACC values with respect to λ and γ. However, we also find that the performance differences on PCMCA, BASEHOCK, and RELATHE are not greater than 0.8%, 0.8%, and 0.4%, respectively. Therefore we cannot make any conclusion on the influence from two balance parameters on ACC based on these 3 datasets. For the parameter λ, which controls the column sparsity of $\mathbf{W}_1$, we can find that for Yale the performance monotonically improves as the value of λ increases for each fixed value of γ, even though the number of selected features m is fixed. We believe this is further evidence that a small number of selected features receiving large score (corresponding to large λ) is sufficient to obtain good learning performance, while having a large number of highly scoring features (corresponding to small λ) may introduce irrelevant features to the selection. We also find a similar behavior for Prostate_GE and Isolet. For both MNIST and COIL20, we can find that the overall performance is best when $\lambda = 10^{-2}$ and both smaller and larger values of λ degrade the performance. This is because the diversity among instances of these two datasets is large enough: a large value of λ may remove informative features, while a small value of λ prevents the exclusion of small, irrelevant, or redundant features. For the parameter γ, which controls local data geometric structure preservation, we can find that both large values and small values of γ degrade performance. On one hand, we can conclude that local data geometric structure preservation does help improve feature selection performance to a certain degree. On the other hand, large weights on local data geometric structure preservation may also harm feature selection performance.



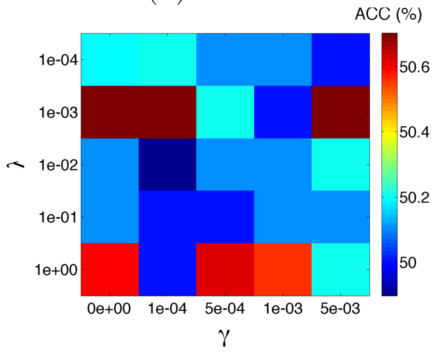
(a) MNIST



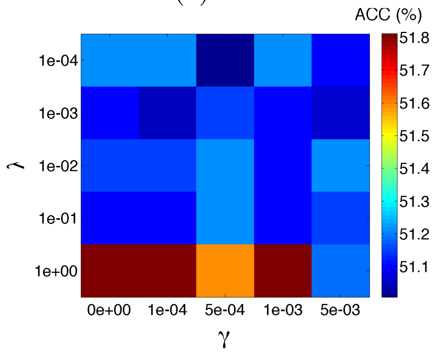
(b) COIL20



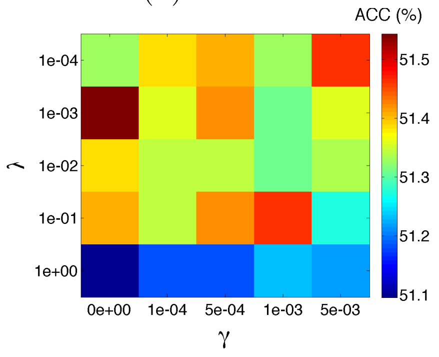
(c) Yale



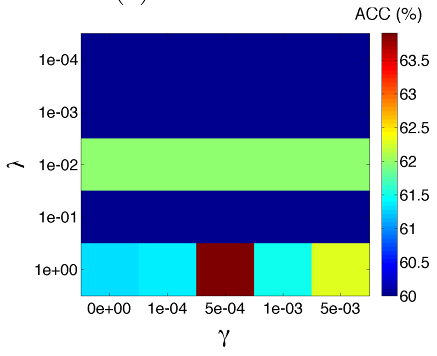
(d) PCMAC



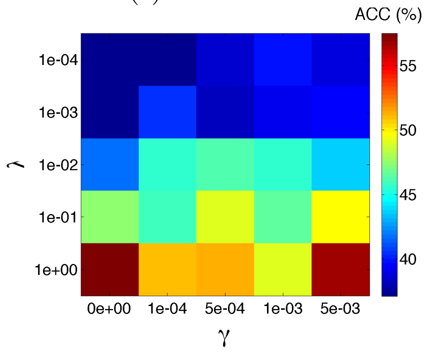
(e) BASEHOCK



(f) RELATHE



(g) Prostate_GE



(h) Isolet

Figure 3.2. Performance variation of the GAFS w.r.t. balance parameters λ and γ.

3.3.4 Feature Selection Illustration

We randomly select five samples from the Yale dataset to illustrate the choices made by different feature selection algorithms. For each sample, $p \in \{10\%, 20\%, 30\%, 40\%, 50\%, 60\%$ features are selected. Figure 3.3 shows images corresponding to the selected features (i.e., pixels) for each sample and value of p, with unselected pixels shown in white. The figure shows that GAFS is able to capture the most discriminative parts on human face such as eyes, nose, and mouse.

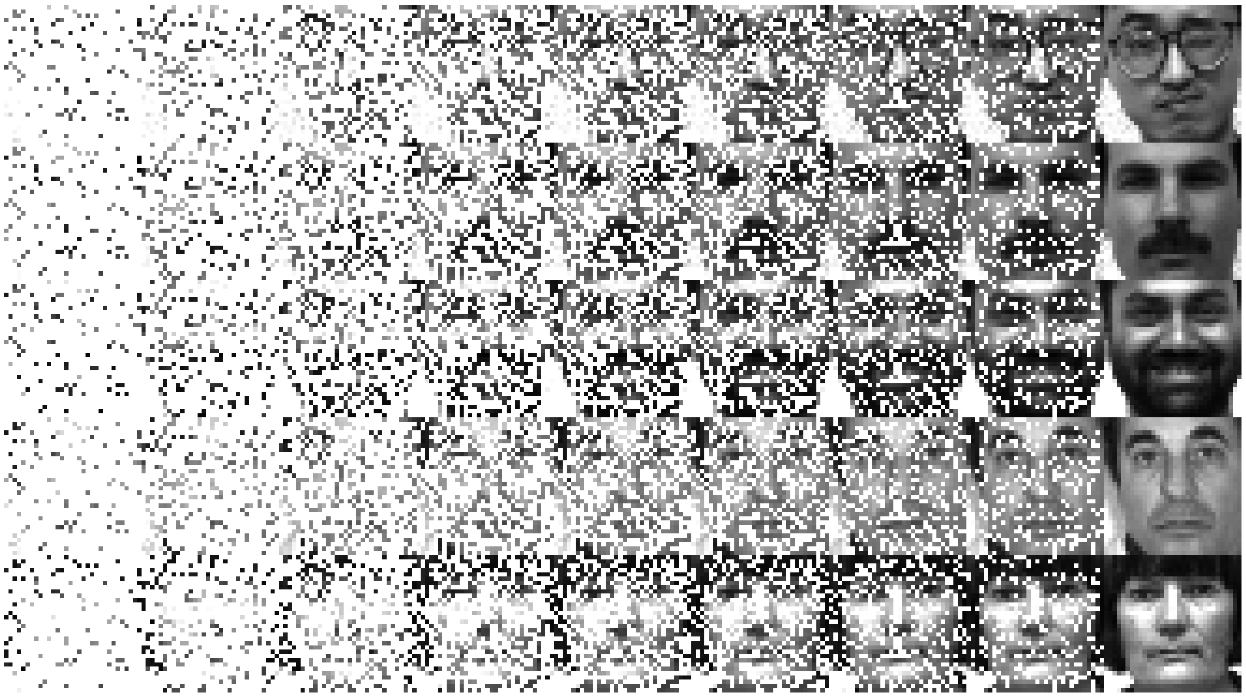
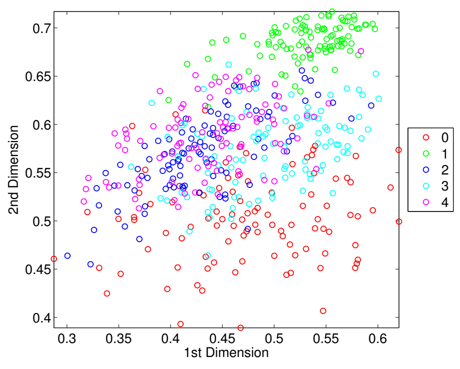


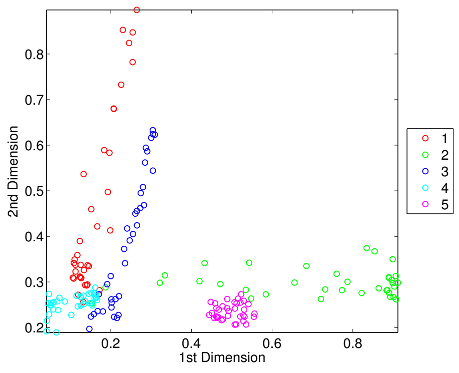
Figure 3.3. Feature selection illustration on Yale. Each row corresponds to a sample human face image and each column refers to percentages of features selected $p \in \{10\%, 20\%, 30\%, 40\%, 50\%, 60\%, 70\%, 80\%, 100\%\}$ from left to right.

3.3.5 Clustering Illustration

We show a toy example of clustering from the low-dimensional data representation (i.e., the hidden layer features) when the hidden layer size is set to $m = 2$ in Fig. 3.4, with circles of different colors representing data embeddings corresponding to different digits for MNIST and different classes for COIL20; we focus on these two datasets



(a) MNIST



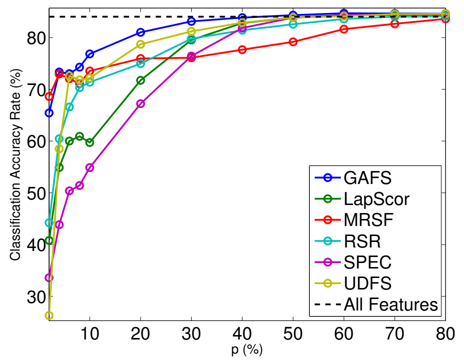
(b) COIL20

Figure 3.4. Clustering illustration of data embeddings on hidden layers when hidden layer size is 2.

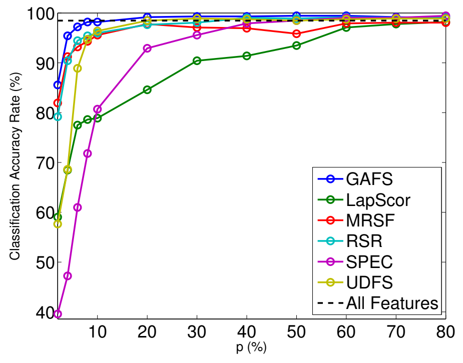
due to space constraints. In this experiment, we empirically set the parameters $\{\lambda = 10^{-2}, \gamma = 10^{-3}\}$ for MNIST and $\{\lambda = 10^{-3}, \gamma = 5\times10^{-3}\}$ for COIL20 for autoencoder training. In order to present a clear illustration, we show the clustering results of digits from 0 to 4 for MNIST instead of 0 to 9 and 5 randomly selected classes for COIL20 instead of all 20 classes, respectively. From Fig. 3.4 we can find that for COIL20 the 5 classes are well separated, while for MNIST the digits 0 and 1 are will separated from other digits.

3.3.6 Performance Comparison

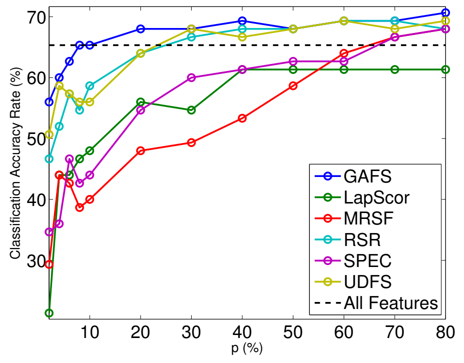
We present the classification accuracy, ACC, and NMI results of GAFS and the comparison feature selection algorithms on all datasets in Fig. 3.5, Fig. 3.6, and Fig. 3.7, respectively. From these figures, we can find that GAFS performs better than other compared algorithms in most cases. Comparing the performance of GAFS with that of using all features, which is represented by a black dashed line in each figure, we can find that GAFS can always achieve better performance with far less features. In the meanwhile, with fewer features, the computational load in corresponding classification and clustering tasks can be decreased. These results demonstrate the effectiveness of GAFS in terms of removing irrelevant and redundant features in classification and clustering tasks.



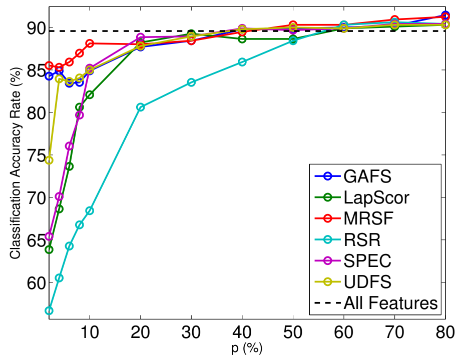
(a) MNIST



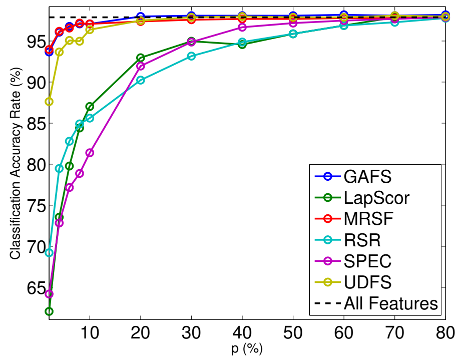
(b) COIL20



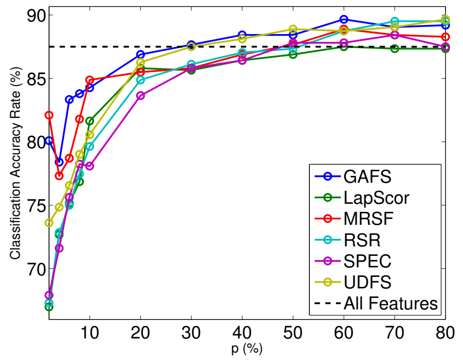
(c) Yale



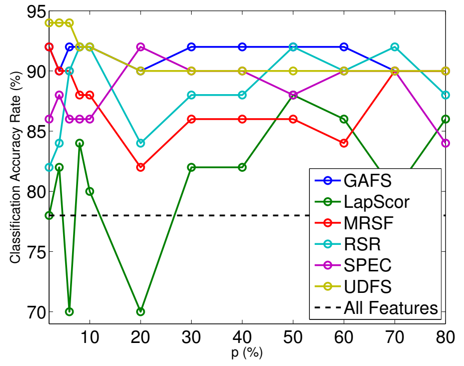
(d) PCMAC



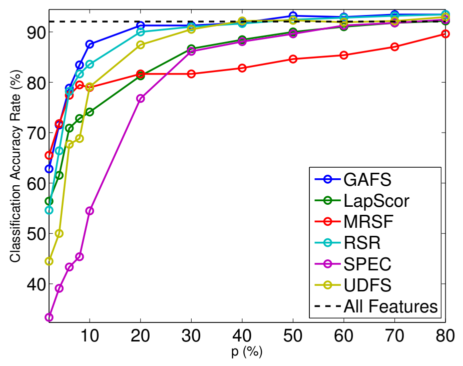
(e) BASEHOCK



(f) RELATHE

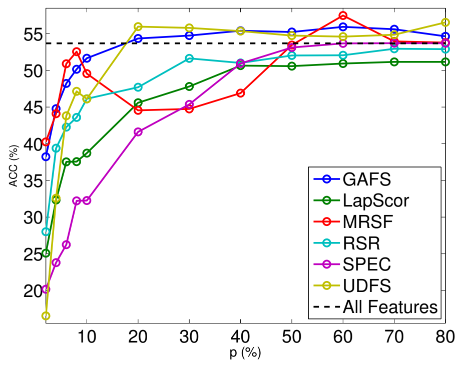


(g) Prostate_GE

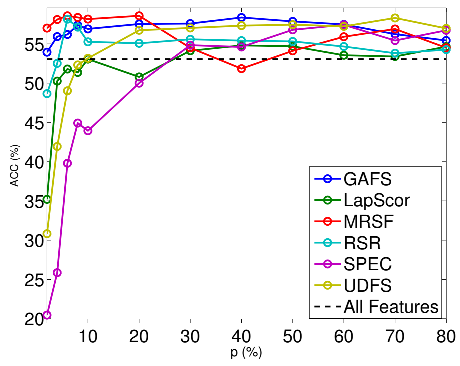


(h) Isolet

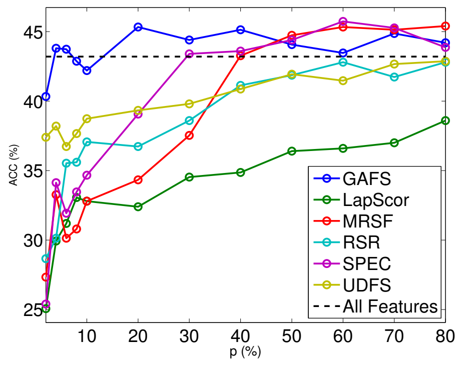
Figure 3.5. Classification accuracy w.r.t different unsupervised feature selection algorithms and the percentage of features selection p (%)



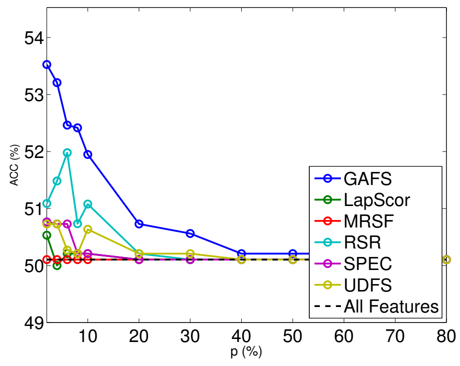
(a) MNIST



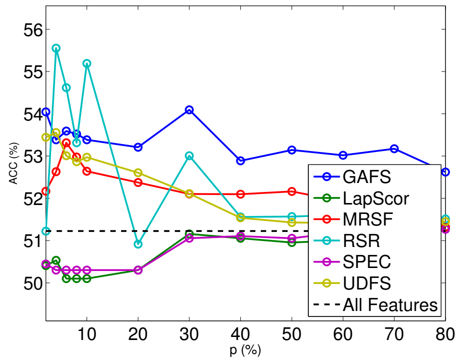
(b) COIL20



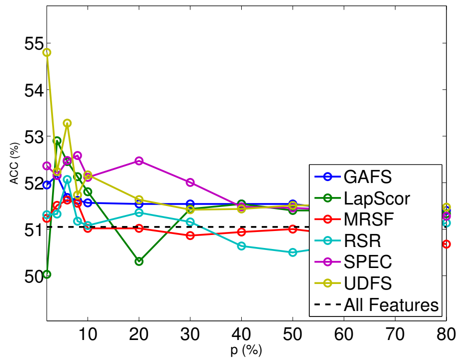
(c) Yale



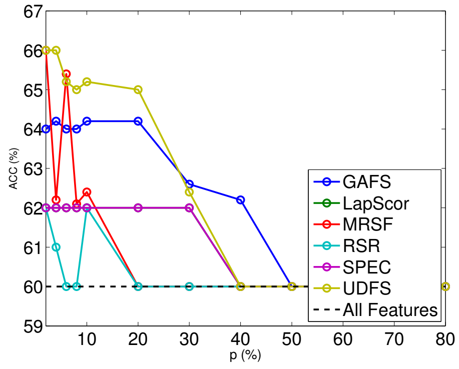
(d) PCMAC



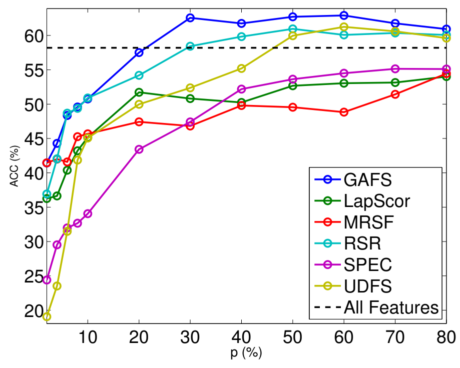
(e) BASEHOCK



(f) RELATHE

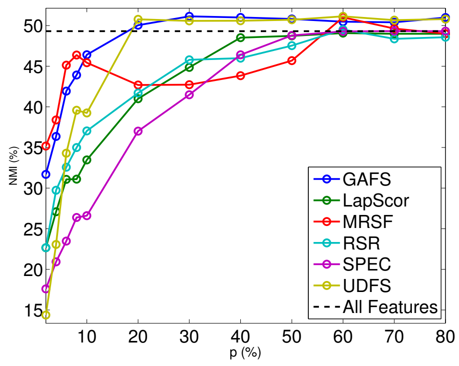


(g) Prostate_GE

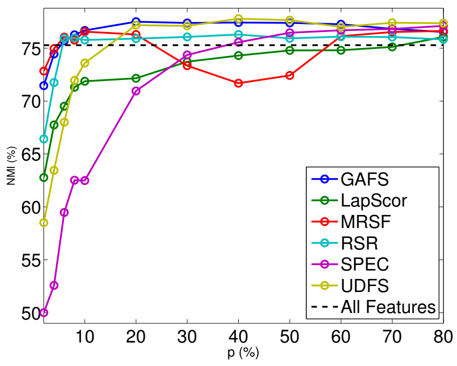


(h) Isolet

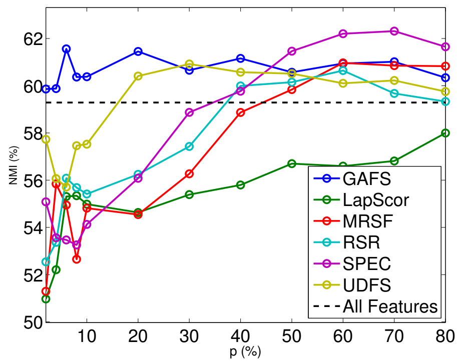
Figure 3.6. Clustering accuracy w.r.t different unsupervised feature selection algorithms and the percentage of features selection p (%)



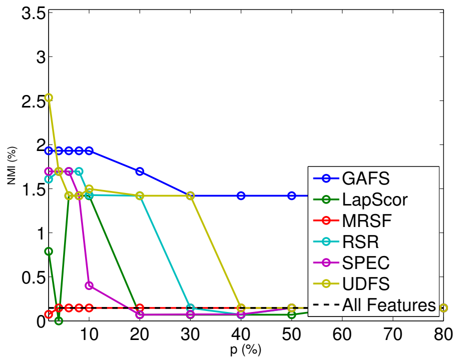
(a) MNIST



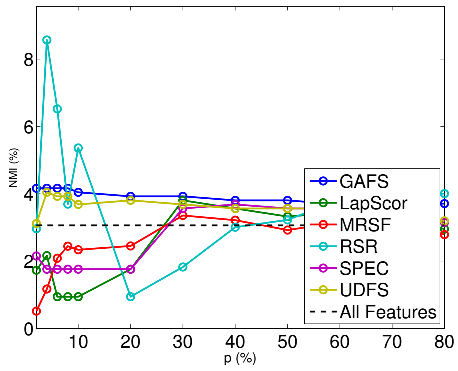
(b) COIL20



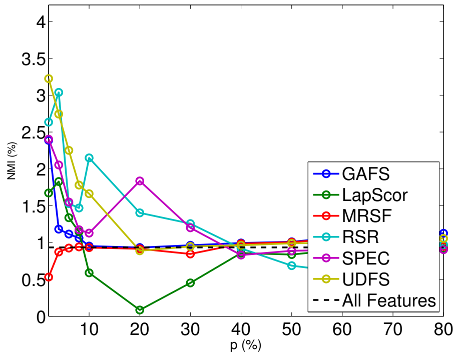
(c) Yale



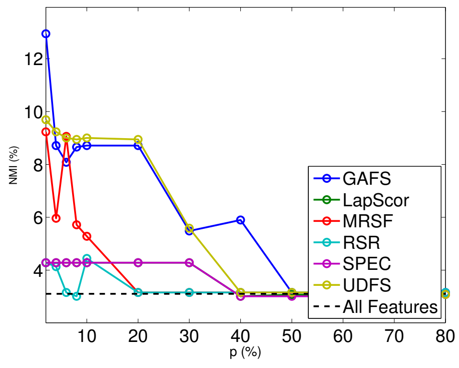
(d) PCMAC



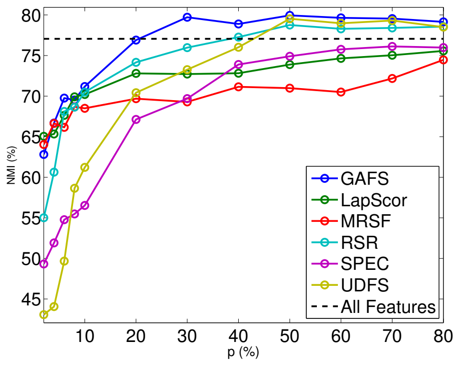
(e) BASEHOCK



(f) RELATHE



(g) Prostate_GE



(h) Isolet

Figure 3.7. Normalized mutual information w.r.t different unsupervised feature selection algorithms and the percentage of features selection p (%)

CHAPTER 4
SELF-TAUGHT LEARNING

4.1 Introduction

Self-taught learning is a technique that uses a large number of unlabeled data as source samples to improve the task performance on target samples. Compared with other transfer learning techniques, self-taught learning can be applied to a broader set of scenarios due to the loose restrictions on source data. However, knowledge transferred from source samples that are not sufficiently related to the target domain may negatively influence the target learner, which is referred to as negative transfer. In this paper, we propose a metric for the relevance between a source sample and target samples. To be more specific, both source and target samples are reconstructed through a single-layer autoencoder with a linear relationship between source samples and target samples simultaneously enforced. An $\ell_{2,1}$-norm sparsity constraint is imposed on the transformation matrix to identify source samples relevant to the target domain. Source domain samples that are deemed relevant are assigned pseudo-labels reflecting their relevance to target domain samples, and are combined with target samples in order to provide an expanded training set for classifier training. Local data structures are also preserved during source sample selection through spectral graph analysis. Promising results in extensive experiments show the advantages of the proposed approach[1].

[1]Contents presented in this chapter are available in [95].

4.2 Proposed Method

In this section, we introduce our proposed GASTL approach. The basic framework of GASTL is to reconstruct both source and target samples through a single-layer autoencoder, while simultaneously enforcing a linear relationship between source samples and target samples. Both global and local data structures are preserved through a single-layer autoencoder and spectral graph analysis, respectively. We develop a metric for the relevance between each source sample and target samples, which is used for source sample selection so that only samples with high relevance are selected for knowledge transfer. Meanwhile, a weight is assigned to each source sample reflecting its relevance to the target samples for the subsequent classifier training, during which each selected source sample is assigned a pseudo-label from the target domain label space and combined with target samples to build the classifier training sample set. Source sample weights are also considered during classifier training. Finally, the trained classifier is used to predict labels of unseen target samples.

4.2.1 Knowledge Transfer and Relevance Measure

In this section we present the problem formulation of our knowledge transfer scheme as well as the corresponding optimization. We also propose a measure for relevance between each source sample and target samples.

4.2.1.1 Objective Function

The objective function of GASTL includes four parts: a data reconstruction term, a domain mapping term, a regularization term for sample selection; and a term based on spectral graph analysis for local data structure preservation. The details of these four terms are described below.

Many transfer learning methods perform knowledge transfer from a source domain to a target domain by finding a mapping between them, that is, $\mathrm{h}_1(\mathbf{X}_{\mathrm{trg}}) = \mathrm{h}_2(\mathbf{X}_{\mathrm{src}})\mathbf{A}$, where $\mathrm{h}_1(\cdot)$ and $\mathrm{h}_2(\cdot)$ are two transformations, while $\mathbf{A}$ is a matrix that linearly maps transformed source samples $\mathrm{h}_2(\mathbf{X}_{\mathrm{src}})$ into transformed target samples $\mathrm{h}_1(\mathbf{X}_{\mathrm{trg}})$. More specifically, the mapping is obtained from the optimization:

$$\min_{\mathbf{\Theta},\mathbf{A}} \mathcal{M}(\mathbf{\Theta},\mathbf{A}) + \lambda\mathcal{R}(\mathbf{A}), \tag{4.1}$$

where $\mathbf{\Theta}$ is a set of parameters used for the nonlinear mappings $\mathrm{h}_1, \mathrm{h}_2$, while $\mathcal{M}(\mathbf{\Theta},\mathbf{A}) = \mathrm{L}\left(\mathrm{h}_1(\mathbf{X}_{\mathrm{trg}}), \mathrm{h}_2(\mathbf{X}_{\mathrm{src}})\mathbf{A}\right)$ denotes a cost function for domain mapping, where $\mathrm{L}(\cdot,\cdot)$ is a loss function and $\mathcal{R}(\cdot)$ corresponds to a regularization function on $\mathbf{A}$ to avoid over-fitting.[2]

A simple way to achieve domain mapping is to assume a linear mapping between source and target data, which is $\mathbf{X}_{\mathrm{trg}} = \mathbf{X}_{\mathrm{src}}\mathbf{A}$. This requires the cost $\mathcal{M}(\mathbf{\Theta},\mathbf{A}) = \mathrm{L}(\mathbf{X}_{\mathrm{trg}}, \mathbf{X}_{\mathrm{src}}\mathbf{A})$. The use of a linear mapping in knowledge transfer is often computationally efficient. However, the success of this knowledge transfer scheme relies on an assumption that $\mathbf{X}_{\mathrm{trg}} \in \mathrm{span}(\mathbf{X}_{\mathrm{src}})$ [96]. Due to the ubiquitous large discrepancy between source and target domain in self-taught learning scenarios, $\mathbf{X}_{\mathrm{trg}}$ is usually not in the span of $\mathbf{X}_{\mathrm{src}}$, and hence a linear reconstruction scheme can hardly do well in knowledge transfer. Therefore, we need to find a non-linear reconstruction scheme that can decrease the discrepancy between source and target domains. One possible way to do this is to find a nonlinear transformation on $\mathbf{X}_{\mathrm{trg}}$, and recover the output of this transformation as a linear transformation of source samples which are relevant to the target samples. That is, $\mathrm{h}(\mathbf{X}_{\mathrm{trg}}) = \mathbf{X}_{\mathrm{src}}\mathbf{A}$, where $\mathrm{h}(\cdot)$ is a nonlinear transformation. Furthermore, due to the possible large diversity of source samples compared with target samples, we can assume that the feature space shared by both source and target domains can be separated into several clusters: the source samples lie near a union of many clusters, while the target samples concentrate near a single cluster.

[2]We empirically found that regularizating $\mathbf{\Theta}$ did not affect the performance of knowledge transfer much. Therefore, we do not pursue such regularization.

Intuitively, negative transfer can be alleviated through using source samples close to target samples for knowledge transfer.

As mentioned in Chapter 2.2, a single-layer autoencoder aims at minimizing the reconstruction error between output and input data. We use $\mathbf{X} = [\mathbf{X}_{\text{src}}\ \mathbf{X}_{\text{trg}}]$ as the input to a single-layer autoencoder by optimizing a reconstruction error-driven loss function:

$$\mathcal{L}(\mathbf{\Theta}) = \frac{1}{2n}\|\mathbf{X} - \mathrm{h}(\mathbf{X};\mathbf{\Theta})\|_F^2, \tag{4.2}$$

where $n = n_{\text{src}} + n_{\text{trg}}$, $\mathbf{\Theta} = [\mathbf{W}_1, \mathbf{W}_2, \mathbf{b}_1, \mathbf{b}_2]$, and $\mathrm{h}(\mathbf{X};\mathbf{\Theta}) = \mathrm{g}\left(\mathbf{W}_2 \cdot \mathrm{f}(\mathbf{W}_1\mathbf{X} + \mathbf{b}_1) + \mathbf{b}_2\right)$.[3] We use the sigmoid function as the activation function: $\mathrm{f}(z) = \mathrm{g}(z) = 1/(1+\exp(-z))$. In Eq. (4.2), both source and target samples share the same parameters to train an autoencoder, which makes the reconstructed source and target samples lie in the same submanifold under the learned parameters $\mathbf{\Theta}$. Meanwhile, we use the following minimization problem for the purpose of domain mapping:

$$\mathcal{C}(\mathbf{\Theta}, \mathbf{A}) = \frac{1}{2n_{\text{trg}}}\|\mathbf{X}_{\text{src}}\mathbf{A} - \mathrm{h}(\mathbf{X}_{\text{trg}};\mathbf{\Theta})\|_F^2. \tag{4.3}$$

That is, we enforce the target samples in the autoencoder output to be reconstructed by a linear combination of the source samples. While it is feasible to separate the optimization of Eq. (4.2) and Eq. (4.3), we observed that a joint framework is able to provide better knowledge transfer performance. Due to the nonlinear nature of transformation featured by a single-layer autoencoder, the distribution gap can be ameliorated through minimizing $\mathcal{C}(\mathbf{\Theta}, \mathbf{A})$ with respect to $\mathbf{\Theta}$ and $\mathbf{A}$. Therefore, we define the mapping cost

$$\mathcal{M}(\mathbf{\Theta}, \mathbf{A}) = \mathcal{L}(\mathbf{\Theta}) + \mu\mathcal{C}(\mathbf{\Theta}, \mathbf{A}), \tag{4.4}$$

[3]We often drop the dependence on $\mathbf{\Theta}$ for readability, i.e. we use $\mathrm{h}(\mathbf{X})$ to denote $\mathrm{h}(\mathbf{X};\mathbf{\Theta})$ when no ambiguity is caused.

where μ is a balance parameter, to obtain a nonlinear mapping between source samples and target samples.

Since each row of $\mathbf{A}$ indicates the importance of the corresponding source sample in reconstrucing transformed target samples, we use the ℓ_2-norm of each row of $\mathbf{A}$ to measure the *relevance* between a source sample and target samples. This leads to an $\ell_{2,1}$-norm regularization function $\mathcal{R}(\mathbf{A}) = \|\mathbf{A}\|_{2,1}$ that enforces row sparsity on the transformation matrix $\mathbf{A}$.

Data transformation based on autoencoders only guarantees broad data structure preservation, which does not take pair-wise relationship between data points into consideration. Therefore, we need to include local data geometric structures into our objective function. Local geometric structures of the data often contain discriminative information of neighboring data point pairs [18]. They assume that nearby data points should have similar representations. In order to characterize the local data structure, we construct a k-nearest neighbor (kNN) graph $\mathbb{G}$ on the data space. The edge weight between two connected data points is determined by the similarity between those two points. We define the adjacency matrix $\mathbf{S}$ for the graph $\mathbb{G}$ as follows: for a data point $\mathbf{X}^{(i)}$, its weight $\mathbf{S}^{(i,j)} \neq 0$ if and only if $\mathbf{X}^{(i)} \in \mathcal{N}_k(\mathbf{X}^{(j)})$ or $\mathbf{X}^{(j)} \in \mathcal{N}_k(\mathbf{X}^{(i)})$, where $\mathcal{N}_k(\mathbf{X}^{(i)})$ denotes the k-nearest neighborhood set for $\mathbf{X}^{(i)}$; otherwise, $\mathbf{S}^{(i,j)} = 0$. We use cosine distance to determine nonzero weight given by $\mathbf{S}^{(i,j)} = (\mathbf{X}^{(i)^T}\mathbf{X}^{(j)})/(\|\mathbf{X}^{(i)}\|_2\|\mathbf{X}^{(j)}\|_2)$. The Laplacian matrix $\mathbf{L}$ of the graph $\mathbb{G}$ is defined as $\mathbf{L} = \mathbf{D} - \mathbf{S}$, where $\mathbf{D}$ is a diagonal matrix whose i^{th} element on the diagonal is defined as $\mathbf{D}^{(i,i)} = \sum_{j=1}^{n} \mathbf{S}^{(i,j)}$. With these definitions, we set up the following minimization objective for local data structure preservation:

$$\begin{aligned}\mathcal{G}(\mathbf{\Theta}) &= \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\|\mathbf{Z}^{(i)} - \mathbf{Z}^{(j)}\|_2^2\mathbf{S}^{(i,j)} \\ &= \sum_{i=1}^{n}\mathbf{Z}^{(i)^T}\mathbf{Z}^{(i)}\mathbf{D}^{(i,i)} - \sum_{i=1}^{n}\sum_{j=1}^{n}\mathbf{Z}^{(i)^T}\mathbf{Z}^{(j)}\mathbf{S}^{(i,j)} \\ &= \mathrm{Tr}(\mathbf{ZDZ}^{\mathrm{T}}) - \mathrm{Tr}(\mathbf{ZSZ}^{\mathrm{T}}) = \mathrm{Tr}(\mathbf{ZLZ}^{\mathrm{T}}),\end{aligned} \tag{4.5}$$

where $\mathbf{Z}^{(i)} = \mathrm{f}(\mathbf{W}_1\mathbf{X}^{(i)} + \mathbf{b}_1)$ for $i = 1, 2, \cdots, n$, and $\mathbf{Z} = [\mathbf{Z}^{(1)}, \mathbf{Z}^{(2)}, \cdots, \mathbf{Z}^{(n)}]$.

The final objective function of source sample selection can be written in terms of the following minimization with respect to the parameters $\mathbf{\Theta} = [\mathbf{W}_1, \mathbf{W}_2, \mathbf{b}_1, \mathbf{b}_2]$ and $\mathbf{A}$:

$$\{\hat{\mathbf{\Theta}}, \hat{\mathbf{A}}\} = \arg\min_{\mathbf{\Theta},\mathbf{A}} \mathcal{L}(\mathbf{\Theta}) + \mu\mathcal{C}(\mathbf{\Theta}, \mathbf{A}) + \lambda\mathcal{R}(\mathbf{A}) + \gamma\mathcal{G}(\mathbf{\Theta}), \tag{4.6}$$

where μ, λ, and γ are balance parameters.

4.2.1.2 Optimization

The closed form solution of the optimization problem in Eq. (4.6) is hard to obtain due to the $\ell_{2,1}$-norm regularization term. We employ an alternating optimization scheme to solve this problem with $\mathbf{\Theta}$ and $\mathbf{A}$ being iteratively updated, until the objective function value in Eq. (4.6) converges or a maximum number of iterations is reached.

When $\mathbf{A}$ is fixed, Eq. (4.6) becomes

$$\hat{\mathbf{\Theta}} = \arg\min_{\mathbf{\Theta}} \mathcal{F}_1(\mathbf{\Theta}) := \arg\min_{\mathbf{\Theta}} \mathcal{L}(\mathbf{\Theta}) + \mu\mathcal{C}(\mathbf{\Theta}, \mathbf{A}) + \gamma\mathcal{G}(\mathbf{\Theta}). \tag{4.7}$$

Following [97], we use a limited memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) algorithm to solve Eq. (4.7). The L-BFGS algorithm has low computational cost, making it possible to use the whole dataset for optimization and provide more stable performance than commonly used stochastic gradient descent algorithms. For example, the dimensionality of the parameter $\mathbf{\Theta}$ is the sum of the dimensionalities of $\mathbf{W}_1 \in \mathbb{R}^{m\times d}$, $\mathbf{W}_2 \in \mathbb{R}^{d\times m}$, $\mathbf{b}_1 \in \mathbb{R}^m$, and $\mathbf{b}_2 \in \mathbb{R}^d$, which is $2md + d + m$. Compared with the conventional BFGS algorithm, which requires computing and storing of $(2md+d+m)\times(2md+d+m)$ Hessian matrices, the L-BFGS algorithm saves the past l updates of $\mathbf{\Theta}$ and corresponding gradients. Therefore, denoting the number of iterations in the optimization by t, the corresponding computational complexity of L-BFGS is $O(tlmd)$. We refer readers to [91] for more details on L-BFGS algorithm, which we implement using the *minFunc* toolbox [92]. The solver requires the gradients of the objective function in Eq. (4.7) with respect to its parameters $\mathbf{\Theta}$. The gradients for both $\mathcal{L}(\mathbf{\Theta})$ and $\mathcal{C}(\mathbf{\Theta},\mathbf{A})$ can be obtained through a back-propagation algorithm. We skip the details for the derivation of the gradients of both $\mathcal{L}(\mathbf{\Theta})$ and $\mathcal{C}(\mathbf{\Theta})$, which are standard in the formulation of backpropagation for an autoencoder. The resulting gradients for $\mathcal{L}(\mathbf{\Theta})$ are:

$$\begin{aligned}\frac{\partial\mathcal{L}(\mathbf{\Theta})}{\partial\mathbf{W}_1} &= \frac{1}{n}\mathbf{\Delta}_{\mathcal{L}2}\mathbf{X}^T, & \frac{\partial\mathcal{L}(\mathbf{\Theta})}{\partial\mathbf{W}_2} &= \frac{1}{n}\mathbf{\Delta}_{\mathcal{L}3}\mathbf{Y}^T,\\ \frac{\partial\mathcal{L}(\mathbf{\Theta})}{\partial\mathbf{b}_1} &= \frac{1}{n}\mathbf{\Delta}_{\mathcal{L}2}\mathbf{1}, & \frac{\partial\mathcal{L}(\mathbf{\Theta})}{\partial\mathbf{b}_2} &= \frac{1}{n}\mathbf{\Delta}_{\mathcal{L}3}\mathbf{1},\end{aligned} \tag{4.8}$$

where each column of $\mathbf{\Delta}_{\mathcal{L}2} \in \mathbb{R}^{m\times n}$ and $\mathbf{\Delta}_{\mathcal{L}3} \in \mathbb{R}^{d\times n}$ contains the error term of the corresponding sample for the hidden layer and the output layer, respectively:

$$\begin{aligned}\mathbf{\Delta}_{\mathcal{L}3} &= (\mathrm{h}(\mathbf{X}) - \mathbf{X}) \bullet \mathrm{h}(\mathbf{X}) \bullet (\mathbf{1} - \mathrm{h}(\mathbf{X})),\\ \mathbf{\Delta}_{\mathcal{L}2} &= (\mathbf{W}_2^T\mathbf{\Delta}_{\mathcal{L}3}) \bullet \mathbf{Y} \bullet (\mathbf{1} - \mathbf{Y}),\end{aligned}$$

with $\bullet$ denoting the element-wise product operator. The gradients for $\mathcal{C}(\mathbf{\Theta},\mathbf{A})$ are:

$$\begin{aligned}\frac{\partial\mathcal{C}(\mathbf{\Theta},\mathbf{A})}{\partial\mathbf{W}_1} &= \frac{1}{n_{\text{trg}}}\mathbf{\Delta}_{\mathcal{C}2}(\mathbf{X}_{\text{src}}\mathbf{A})^T, & \frac{\partial\mathcal{C}(\mathbf{\Theta},\mathbf{A})}{\partial\mathbf{W}_2} &= \frac{1}{n_{\text{trg}}}\mathbf{\Delta}_{\mathcal{C}3}\mathbf{Y}_{\text{trg}}^T,\\ \frac{\partial\mathcal{C}(\mathbf{\Theta},\mathbf{A})}{\partial\mathbf{b}_1} &= \frac{1}{n_{\text{trg}}}\mathbf{\Delta}_{\mathcal{C}2}\mathbf{1}, & \frac{\partial\mathcal{C}(\mathbf{\Theta},\mathbf{A})}{\partial\mathbf{b}_2} &= \frac{1}{n_{\text{trg}}}\mathbf{\Delta}_{\mathcal{C}3}\mathbf{1}.\end{aligned} \tag{4.9}$$

Both $\mathbf{\Delta}_{\mathcal{L}2}^{(i)}$ and $\mathbf{\Delta}_{\mathcal{L}3}^{(i)}$ in Eq. (4.8) play same roles as $\mathbf{\Delta}_{\mathcal{C}2}^{(i)}$ and $\mathbf{\Delta}_{\mathcal{C}3}^{(i)}$ in Eq. (4.9). Their definitions are:

$$\begin{aligned}
\boldsymbol{\Delta}_{\mathcal{C}3} &= (\mathrm{h}(\mathbf{X}_{\mathrm{trg}}) - \mathbf{X}_{\mathrm{src}}\mathbf{A}) \bullet \mathrm{h}(\mathbf{X}_{\mathrm{trg}}) \bullet (\mathbf{1} - \mathrm{h}(\mathbf{X}_{\mathrm{trg}})),\\
\boldsymbol{\Delta}_{\mathcal{C}2} &= (\mathbf{W}_2^T \boldsymbol{\Delta}_{\mathcal{C}3}) \bullet \mathbf{Y}_{\mathrm{trg}} \bullet (\mathbf{1} - \mathbf{Y}_{\mathrm{trg}}),
\end{aligned}$$

where $\mathbf{Y}_{\mathrm{trg}} = \mathrm{f}(\mathbf{W}_1\mathbf{X}_{\mathrm{trg}} + \mathbf{b}_1)$. The gradients of the graph term $\mathcal{G}(\boldsymbol{\Theta}) = \mathrm{Tr}(\mathbf{YLY}^{\mathrm{T}})$ can be obtained in a straightforward fashion as follows:

$$\begin{aligned}
\frac{\partial \mathcal{G}(\boldsymbol{\Theta})}{\partial \mathbf{W}_1} &= \frac{\partial \mathrm{Tr}(\mathbf{YLY}^{\mathrm{T}})}{\partial \mathbf{Y}} \cdot \frac{\partial \mathbf{Y}}{\partial \mathbf{W}_1} = 2\left(\mathbf{YL} \bullet \mathbf{Y} \bullet (\mathbf{1} - \mathbf{Y})\right)\mathbf{X}^T,\\
\frac{\partial \mathcal{G}(\boldsymbol{\Theta})}{\partial \mathbf{W}_2} &= \mathbf{0},\\
\frac{\partial \mathcal{G}(\boldsymbol{\Theta})}{\partial \mathbf{b}_1} &= \frac{\partial \mathrm{Tr}(\mathbf{YLY}^{\mathrm{T}})}{\partial \mathbf{Y}} \cdot \frac{\partial \mathbf{Y}}{\partial \mathbf{b}_1} = 2\left(\mathbf{YL} \bullet \mathbf{Y} \bullet (\mathbf{1} - \mathbf{Y})\right)\mathbf{1},\\
\frac{\partial \mathcal{G}(\boldsymbol{\Theta})}{\partial \mathbf{b}_2} &= \mathbf{0}.
\end{aligned}$$

To conclude, the gradients of the objective function in Eq. (4.7) with respect to $\boldsymbol{\Theta} = [\mathbf{W}_1, \mathbf{W}_2, \mathbf{b}_1, \mathbf{b}_2]$ can be written as

$$\begin{aligned}
\frac{\partial \mathcal{F}_1(\boldsymbol{\Theta})}{\partial \mathbf{W}_1} &= \frac{1}{n}\boldsymbol{\Delta}_{\mathcal{L}2}\mathbf{X}^T + \frac{\mu}{n_{\mathrm{trg}}}\boldsymbol{\Delta}_{\mathcal{C}2}(\mathbf{X}_{\mathrm{src}}\mathbf{A})^T + 2\gamma\left(\mathbf{YL} \bullet \mathbf{Y} \bullet (\mathbf{1} - \mathbf{Y})\right)\mathbf{X}^T,\\
\frac{\partial \mathcal{F}_1(\boldsymbol{\Theta})}{\partial \mathbf{W}_2} &= \frac{1}{n}\boldsymbol{\Delta}_{\mathcal{L}3}\mathbf{Y}^T + \frac{\mu}{n_{\mathrm{trg}}}\boldsymbol{\Delta}_{\mathcal{C}3}\mathbf{Y}_{\mathrm{trg}}^T,\\
\frac{\partial \mathcal{F}_1(\boldsymbol{\Theta})}{\partial \mathbf{b}_1} &= \frac{1}{n}\boldsymbol{\Delta}_{\mathcal{L}\mathbf{2}}\mathbf{1} + \frac{\mu}{n_{\mathrm{trg}}}\boldsymbol{\Delta}_{\mathcal{C}\mathbf{2}}\mathbf{1} + 2\gamma\left(\mathbf{YL} \bullet \mathbf{Y} \bullet (\mathbf{1} - \mathbf{Y})\right)\mathbf{1},\\
\frac{\partial \mathcal{F}_1(\boldsymbol{\Theta})}{\partial \mathbf{b}_2} &= \frac{1}{n}\boldsymbol{\Delta}_{\mathcal{L}\mathbf{3}}\mathbf{1} + \frac{\mu}{n_{\mathrm{trg}}}\boldsymbol{\Delta}_{\mathcal{C}\mathbf{3}}\mathbf{1}.
\end{aligned}$$

When $\boldsymbol{\Theta}$ is fixed, Eq. (4.6) becomes

$$\hat{\mathbf{A}} = \arg\min_{\mathbf{A}} \mathcal{F}_2(\mathbf{A}) := \arg\min_{\mathbf{A}} \mu\mathcal{C}(\boldsymbol{\Theta}, \mathbf{A}) + \lambda\mathcal{R}(\mathbf{A}). \tag{4.10}$$

Following [23], we use the scheme described below to optimize $\mathbf{A}$. The regularization term $\mathcal{R}(\mathbf{A}) = \|\mathbf{A}\|_{2,1}$ and its derivative do not exist for its i^{th} column $\mathbf{A}^{(i)}$ when $\mathbf{A}^{(i)} = \mathbf{0}$. In this case, we calculate the values of elements in $\mathbf{AU}$ to approximate the derivative of $\mathcal{R}(\mathbf{A})$ with respect to $\mathbf{A}$, where $\mathbf{U} \in \mathbb{R}^{d\times d}$ is a diagonal matrix whose ith element on the diagonal is

$$\mathbf{U}^{(i,i)} = \begin{cases} \left(\|\mathbf{A}^{(i)}\|_2 + \epsilon\right)^{-1}, & \|\mathbf{A}^{(i)}\|_2 \neq 0, \\ 0, & \text{otherwise}, \end{cases} \tag{4.11}$$

where ϵ is a small constant added to avoid overflow. Since $\mathcal{R}(\mathbf{A}) = \|\mathbf{A}\|_{2,1}$ is not differentiable when the ℓ_2-norm of a certain row in $\mathbf{A}$ is 0, we calculate the subgradient for each element in $\mathbf{A}$ for that case. That is, for each element in $\mathbf{A}$, the subgradient at 0 can be an arbitrary value in the interval $[-1, 1]$, and so we set the gradient to 0 for computational convenience. In this way, the subgradient of $\mathcal{F}_2(\mathbf{A})$ with respect to $\mathbf{A}$ is:

$$\frac{\partial \mathcal{F}_2(\mathbf{A})}{\partial \mathbf{A}} = \frac{\mu}{n_{\text{trg}}}[\mathbf{X}_{\text{src}}^T\mathbf{X}_{\text{src}}\mathbf{A} - \mathbf{X}_{\text{src}}^T \mathrm{h}(\mathbf{X}_{\text{trg}})] + \lambda\mathbf{A}\mathbf{U}. \tag{4.12}$$

That is, when $\mathbf{U}$ is fixed, an optimal value of $\mathbf{A}$ can be obtained through

$$\hat{\mathbf{A}} = (\mu\mathbf{X}_{\text{src}}^T\mathbf{X}_{\text{src}} + n_{\text{trg}}\lambda\mathbf{U})^{-1}\mu\mathbf{X}_{\text{src}}^T\mathrm{h}(\mathbf{X}_{\text{trg}}). \tag{4.13}$$

Therefore, we can update $\mathbf{U}$ through Eq. (4.11) when $\mathbf{A}$ is fixed and update $\mathbf{A}$ through Eq. (4.13) when $\mathbf{U}$ is fixed with an iterative scheme until the value of $\mathcal{F}_2(\mathbf{A})$ converges.

4.2.2 Classifier Training

The next step is to use source samples combined with target samples to train a classifier, which can then be applied to unseen samples in the target domain for classification. Since source samples have different relevance levels with the target domain, we propose a scheme to assign weights to source samples that reflect their relevance. Source domain samples with large weights are kept while others are discarded. Subsequently, a classifier is trained using both target samples and selected source samples.

For each source sample, pseudo-labels that indicate the transferability of the source sample to different target classes are used as true labels during classifier training, where transferability [98] reflects the possibility of transfering a source sample to a target domain class. The transferability values of source samples are stored in a matrix $\mathbf{Tr} \in \mathbb{R}^{n_{\text{src}} \times n_{\text{ctrg}}}$, where n_{ctrg} is the cardinality of $\mathcal{Y}_{\text{trg}}$. Two pseudo-labeling schemes are proposed for comparison. Additionally, source sample weights are taken into consideration in classifier training. For each pseudo-labeling scheme, we evaluate both soft and hard classification with the softmax classifier.

4.2.2.1 Source Domain Sample Reweighting

As mentioned in Chapter 4.2.1.1, the ℓ_2-norm value of each row of $\mathbf{A}$ can be used to measure the relevance between the corresponding source sample and target samples. We propose a scheme to assign a weight to each source sample based on the corresponding row in $\mathbf{A}$. The weight for a source sample $\mathbf{X}_{\text{src}}^{(i)}|_{i=1}^{n_{\text{src}}}$ is set as the ℓ_2-norm value of the corresponding row in $\mathbf{A}$. That is, for a source sample $\mathbf{X}_{\text{src}}^{(i)}$, its weight vector $\mathbf{Wt} \in \mathbb{R}^{n_{\text{src}}}$ has entries $\mathbf{Wt}(i) = \|\mathbf{A}_{(i)}\|_2 / \max_j \left(\|\mathbf{A}_{(j)}\|_2\right)$. Note that the vector $\mathbf{Wt}$ is normalized with the maximum entry value being 1. In addition, during classifier training, all target training samples are given weight 1.

4.2.2.2 Pseudo-Labeling

Two transferability measure schemes are proposed and pseudo-labels are assigned to source samples based on transferability values.

Scheme A: For a given source sample $\mathbf{X}_{\text{src}}^{(i)}$, its transferability to a target class $\mathbf{c}^{(j)}$ is measured by the square of the ℓ_2-norm of a subvector consisting of elements in the corresponding row of $\mathbf{A}$ that belong to target samples of $\mathbf{c}^{(j)}$. That is, $\mathbf{Tr}^{(i,\mathbf{c}^{(j)})} = \|\mathbf{A}^{(i,\mathbf{J}_{\mathbf{c}^{(j)}})}\|_2^2$, where $\mathbf{J}_{\mathbf{c}^{(j)}}$ denotes the columns in $\mathbf{A}$ that correspond to class $\mathbf{c}^{(j)}$.

Scheme B: By following [98], we adopt the isometric Gaussian probability [99] computed on the hidden layer representation of the trained single-layer autoencoder as the transferability of a given source sample $\mathbf{X}_{\text{src}}^{(i)}$ to a target class $\mathbf{c}^{(j)}$. More concretely, the transferability is

$$\mathbf{Tr}^{(i,\mathbf{c}^{(j)})} = \mathcal{N}(\mathbf{Z}_{\text{src}}^{(i)} | \bar{\mathbf{Z}}_{\text{trg}}^{\mathbf{c}^{(j)}}, \sigma^2 \mathbf{I}), \tag{4.14}$$

where $\mathbf{Z}_{\text{src}}^{(i)} \in \mathbb{R}^m$ is the hidden layer representation of the source sample $\mathbf{X}_{\text{src}}^{(i)}$, and $\bar{\mathbf{Z}}_{\text{trg}}^{\mathbf{c}^{(j)}} \in \mathbb{R}^m$ is the mean of the hidden layer representation of target samples belonging to class $\mathbf{c}^{(j)}$. As such, the transferability is measured by the probability that the source sample belongs to a target class given the auxiliary information $\bar{\mathbf{Z}}_{\text{trg}}^{\mathbf{c}^{(j)}}$.

The pseudo-labels of source samples consist of a matrix $\mathbf{L} \in \mathbb{R}^{n_{\text{src}} \times n_{\text{ctrg}}}$. We assign pseudo-labels to source samples based on their transferability values to different target domain classes. Given a source sample $\mathbf{X}_{\text{src}}^{(i)}$, for a **hard classifier**, we set $\mathbf{L}^{(i,j)} = 1$ if target class $\mathbf{c}^{(j)}$ provides the largest transferability value; otherwise $\mathbf{L}^{(i,j)} = 0$. For a **soft classifier**, the normalized transferability values are used as pseudo-labels so that pseudo lables reflect the likelihood of transfering source samples to target domain classes: $\mathbf{L}^{(i,j)} = \mathbf{Tr}^{(i,\mathbf{c}^{(j)})} / \sum_{k=1}^{n_{\text{ctrg}}} \mathbf{Tr}^{(i,\mathbf{c}^{(k)})}$. Compared with hard classifiers, soft classifiers may help improve knowledge transfer performance since it is able to capture the relationship between each single source sample and multiple target categories instead of one. This is especially necessary for image classification tasks since there usually exists commonalities between image categories.

4.2.2.3 Classifier Training

We employ softmax classfier due to its simplicity and capability to do soft classification. The training data weights are included in classifier training, which leads to the following cost function:

$$\mathrm{J}(\mathbf{\Theta_c}) = -\frac{1}{n}\left[\sum_{i=1}^{n}\mathbf{Wt}(i)\sum_{j=1}^{n_{\text{ctrg}}}\mathbf{L}^{(i,j)}\log\frac{e^{\mathbf{\Theta_c}^{(j)T}\mathbf{X}^{(i)}}}{\sum_{l=1}^{n_{\text{ctrg}}}e^{\mathbf{\Theta_c}^{(l)T}\mathbf{X}^{(i)}}}\right],$$

where $\mathbf{\Theta_c} = [\mathbf{\Theta_c}^{(1)}, \mathbf{\Theta_c}^{(2)}, \cdots, \mathbf{\Theta_c}^{(n_{\text{ctrg}})}]$ is the classifier parameter to be optimized. We use an L-BFGS algorithm to compute the optimal value of $\mathbf{\Theta}$. The gradients needed for optimization are given by

$$\frac{\partial \mathrm{J}(\mathbf{\Theta_c})}{\partial \mathbf{\Theta_c}^{(j)}} = -\frac{1}{n}\sum_{i=1}^{n}\left[\mathbf{Wt}(i)\mathbf{X}^{(i)}\left(\mathbf{L}^{(i,j)} - \frac{e^{\mathbf{\Theta_c}^{(j)T}\mathbf{X}^{(i)}}}{\sum_{l=1}^{n_{\text{ctrg}}}e^{\mathbf{\Theta_c}^{(l)T}\mathbf{X}^{(i)}}}\right)\right].$$

4.3 Experiments

In this section, we evaluate the knowledge transfer performance of GASTL. Experiments are conducted on four benchmark datasets covering computer vision, natural language processing, and speech recognition. We also compare GASTL with other relevant state-of-the-art transfer learning techniques. To be more specific, we first select p source samples which are the most relevant to the target domain, and then use those selected source samples combined with labeled target samples to train a classifier. The classification rates on target testing samples are then used as metric to evaluate knowledge transfer performance.

4.3.1 Dataset Preparation

We introduce the information of datasets we use in our experiments. We first provide the overall information of each dataset. After that we introduce the source/target domain setup.

- **Dataset Information:** We employ four benchmark datasets in our experiments, including one visual dataset (Caltech101[4]), two natural language datasets (IMDB[5] and Twitter,[6] both for sentiment analysis), and one audio dataset (ESC-50[7]). In order to eliminate the side effects caused by imbalanced classes, we set the number of samples from each class to be the same within each dataset through random selection. For Caltech101, we keep 30 images for each class. For both IMDB and Twitter, we only consider texts with no more than 10 words. The ESC-50 dataset already has an even class setup with each class containing 40 audio clips. The properties of these datasets are summarized in Table 4.1.

- **Feature Extraction:** Data in raw feature space cannot be used for knowledge transfer due to possible dimensionality inconsistence. For example, images may have different sizes and texts may have different lengths, Therefore, it is necessary to do feature extraction on each dataset to make knowledge transfer feasible. For Caltech101, we employ two types of features. The first one is the 1,000-dimensional SIFT-BOW feature[8] proposed in Gehler et. al. [100]. The second one is the 4,096-dimensional output of the last fully connected layer of the pre-trained VGG-19 model [101]. In our experiments we use the Keras tool[9] to compute the VGG-19

[4]Dataset downloaded from: `http://www.vision.caltech.edu/Image_Datasets/Caltech101/`. The Caltech101 dataset contains both a "Faces" and "Faces easy" class, with each consisting of different versions of the same human face images. However, the images in "Faces" contain more complex backgrounds. To avoid confusion between these two similar classes of images, we do not include the "Faces easy" images in our experiments. Therefore, we keep 100 classes for Caltech101.

[5]Dataset downloaded from: `https://drive.google.com/file/d/0B8yp1gOBCztyN0JaMDVoeXhHWm8/`.

[6]Dataset downloaded from: `https://www.kaggle.com/c/twitter-sentiment-analysis2/data`

[7]Dataset downloaded from: `https://github.com/karoldvl/ESC-50`

[8]Dataset downloaded from: `http://files.is.tue.mpg.de/pgehler/projects/iccv09/`.

[9]We refer readers to `https://github.com/keras-team/keras` for more information on Keras.

Dataset	Features	Samples	Classes	Type
Caltech101 (SIFT-BOW)	1,000	3,000	100	Image
Caltech101 (VGG-19)	4,096	3,000	100	Image
IMDB	3,000	6,500	2	Text
Twitter	3,000	6,500	2	Text
ESC-50	2,592	2,000	50	Audio

Table 4.1. Details of datasets used in our experiment.

features.[10] For both IMDB and Twitter, we use the method from [102][11] to do feature extraction. By using a convolutional neural network model with publicly available WORD2VEC vectors, we get a 300×10 matrix for each text, which forms a 3,000-dimensional feature vector after column concatenation. We denote this feature as WORD2VEC in the sequel. For ESC-50, by following [103], we use the librosa package [104] to compute mel-frequency cepstral coefficients (MFCC) for each clip. After discarding the 0th coefficient, the first 12 MFCCs are whitened and used as feature for each clip. Therefore, we get a 12×216 matrix for each clip,[12] which forms a 2,592-dimensional vector after column concatenation. These 2,592-dimensional vectors are used as features for audio clips. We denote this feature as MFCC in the sequel.

- **Source/Target Split:** For Caltech101, we randomly separate the 100 classes into 5 groups with 20 classes in each group. Five independent self-taught learning experiments were conducted on Caltech101: in each experiment samples in one group are used as target samples and those in the remaining four groups are used as source samples. For each class in the target domain, 15 samples were used for training and 15 samples were used for testing. For the two natural language datasets, we

[10]In the sequel, we use VGG-19 to denote the feature generated by VGG-19 models.

[11]Codes downloaded from: `https://github.com/yoonkim/CNN_sentence`

[12]Each MFCC contains 216 frames in our experiments.

used one as source and the other one as target. That is, when IMDB was used as target, then Twitter was used as source and vice versa. For computational convenience, we did not use the entire datasets when either IMDB and Twitter is used as the source. We randomly selected 3,000 samples as source for both IMDB and Twitter. For each class in the target domain, 10, 100, 1000 samples were used for training and 750 samples were used for testing. The audio clips in ESC-50 dataset are separated into 5 groups with each group containing 10 classes. The 5 groups are: animal sounds; natural soundscapes and water sounds; human, non-speech sounds; interior/domestic sounds; and exterior/urban noises. Like Caltech101, we conducted five independent self-taught learning experiments on ESC-50 that in each experiment samples in one category were used as target and those in the other four categories were used as source. For each class in the target domain of ESC-50, 5, 10, 15, 20 samples were used for training and 20 samples were used for testing.

4.3.2 Experimental Setup

We performed classification on the target testing samples in order to evaluate the effectiveness of the self-taught learning algorithms and two sample selection/re-weighting based domain adaptation methods. The three self-taught learning methods are STL [45], RDSTL [52], and S-Low [53] introduced in Chapter 2.5. The two domain adaptation methods are kernel mean matching (KMM) [105] and multiscale landmarks selection (MLS) [106]. These two domain adaptation methods were tailored to the scenario of self-taught learning. We also compute the classification performance without knowledge transfer.

Both GASTL and the compared algorithms include parameters to adjust. In this experiment, we fix some parameters and tune others through a grid search strategy. For algorithms requiring source sample selection/re-weighting, we select the number of source samples $p \in \{10, 20, 30, \cdots, 100, 150, 200, 250, \cdots, 500, 1000, 1500, n_{\text{src}}\}$.

In GASTL, the range of hidden layer sizes is set to $m \in \{10, 50, 100, 200\}$, while the balance parameters are given ranges of $\lambda \in \{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1\}$ and $\gamma \in \{0, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$. The value of μ is set to 1. The number of nearest neighbor in a kNN graph is set to 5. The value of σ^2 is set to 1.

All three self-taught learning methods (STL, RDSTL, and S-Low) are based on dictionary learning, and sparse code vectors were used as features for classification. For STL, we first performed PCA on each training sample since the features listed in Table 4.1 have high dimensionalities and require large dictionary sizes, which would cause prohibitive training time. Following [45], we kept the number of principal components to preserve approximately 96% of the training sample variance. We also found empirically that small dictionary sizes provide poor performance and large dictionary sizes result in unfeasibly long training times. In this experiment, we empirically set the dictionary size as the training set size, which are 2,400 for Caltech101, 3,000 for both IMDB and Twitter, and 1,600 for ESC-50, for their acceptable training time and good performance.

The balance parameter β in Eq. (4) and Eq. (2.11) is sampled from $\beta \in \{10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$. For both RDSTL and S-Low, we observed that their performance was not very sensitive to dictionary sizes but quite dependent on balance parameters. Therefore, we use the following parameter adjustment scheme. For RDSTL, we first use grid search for parameter β with the range $\{10^{-9}, 10^{-8}, \cdots, 10^{-1}, 1\}$. The parameter value leading to the highest classification accuracy, denoted by $\hat{\beta}_1$, is kept for the next round. Assume $\hat{\beta}_1 = a_1 \times 10^{b_1}$, then the second round of grid search is of the range $\{a_1 - 0.4, \cdots, a_1 - 0.1, a_1 + 0.1, \cdots, a_1 + 0.5\} \times 10^{b_1}$. The parameter value that resulted in the best classification performance in the second round is denoted as $\hat{\beta}_2$. Assume $\hat{\beta}_2 = a_2 \times 10^{b_2}$, then the third round of grid search is of the range $\{a_2 - 0.04, \cdots, a_2 - 0.01, a_2 + 0.01, \cdots, a_2 + 0.05\} \times 10^{b_2}$. The parameter value that resulted in the best classification performance in the third round is denoted as

$\hat{\beta}_3$, which is the final optimal β value. For S-Low, both λ_1 and λ_2 are set to 0.02 and we also use a three-round scheme to adjust the value of λ_3. The initial range of λ_3 is $\{10^{-5}, 10^{-4}, \cdots, 10^{-1}, 1\}$. Due to the large time consumption during parameter adjusting, we also perform PCA with the same scheme that are used for STL.

For both KMM[13] and MLS[14], we first obtained weights for the source samples, which are also assigned pseudo-labels. Between the two pseudo-labeling schemes described in Chapter 4.2.2, only Scheme B is applicable since Scheme A is dependent on the transformation matrix $\mathbf{A}$, and these two domain adaptation methods do not generate one. We use the features listed in Table 4.1 in Eq. (4.14) instead of autoencoder activations as we did in GASTL. Subsequent steps were exactly the same as those for GASTL. For the optimization of GASTL with L-BFGS, we set the number of iterations t to be 400 and the number of storing updates l to be 100.

4.3.3 Parameter Sensitivity

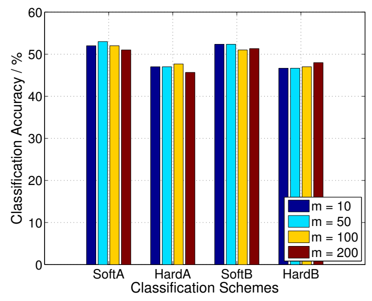
We study the performance variation of GASTL with respect to the hidden layer size m and the two balance parameters λ and γ reflected by classification accuracy on target testing samples. We show the results on all four datasets.

We first study the parameter sensitivity of GASTL with respect to the hidden layer size m. Due to limited space, we only present a small portion of our experimental results in Fig. 4.1,[15] where "Soft" and "Hard" refer to whether a soft or hard classifier is used, while "A" and "B" refer to the pseudo-labeling schemes. Note that although

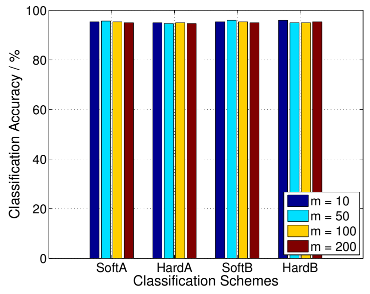
[13]Codes downloaded from: `http://www.gatsby.ucl.ac.uk/~gretton/covariateShiftFiles/covariateShiftSoftware.html`

[14]Codes downloaded from: `https://github.com/jindongwang/transferlearning/tree/master/code`

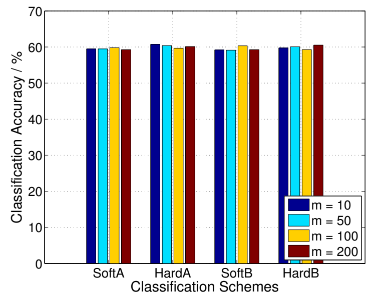
[15]For Caltech101, we use the results of one set out of five with SIFTBOW features as antoencoder inputs. For both IMDB and Twitter, we use the results with each target data having 10 training samples. For ESC-50, we use the results of one set out of five with each target data having 10 training samples.



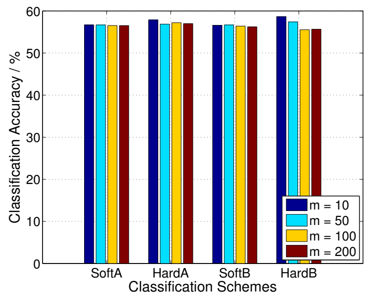
(a) Caltech101 (SIFTBOW)



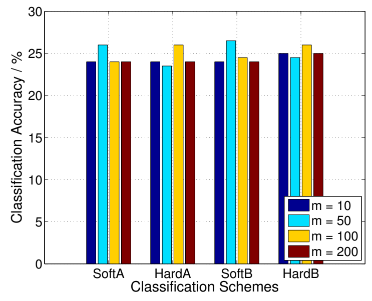
(a) Caltech101 (VGG19)



(b) IMDB



(c) Twitter



(d) ESC-50

Figure 4.1. Performance of GASTL in classification as a function of the hidden layer size m for varying sizes of the autoencoder hidden layer m. Classification accuracy (%) is used as the evaluation metric.

we train classifiers with multiple choices of source sample numbers, the classification results in Fig. 4.1 are the highest classification accuracy among all available choices. The results show that the performance of GASTL is not too sensitive to hidden layer size on the given datasets.

We also study the parameter sensitivity of GASTL with respect to the balance parameters λ and γ, under a fixed hidden layer size. In order to do this, with all other parameters being fixed, we record the classification accuracy corresponding to

Scheme \ Dataset	Caltech101 (SIFT-BOW)	Caltech101 (VGG-19)	IMDB	Twitter	ESC-50
SoftA	48.92 ± 1.53	94.48 ± 0.47	58.22 ± 0.60	55.79 ± 0.47	21.54 ± 1.38
HardA	44.20 ± 1.17	92.84 ± 1.45	57.91 ± 0.98	55.41 ± 1.13	20.06 ± 2.15
SoftB	48.75 ± 1.65	94.44 ± 0.51	58.08 ± 0.54	55.76 ± 0.47	21.68 ± 1.46
HardB	44.27 ± 1.45	94.39 ± 1.20	57.79 ± 0.86	55.06 ± 1.29	20.58 ± 2.11

Table 4.2. Performance stability of GASTL in classification with respect to balance parameters λ and γ. Classification accuracy mean (%) and standard deviation (%) are presented.

each parameter combination, which consists of a 5×5 matrix. We then calculate the mean and standard deviation of these 25 elements, and parameter sensitivity can be evaluated through the ratio between standard deviation and mean value. We choose hidden layer size $m = 10$ as Fig. 4.1 shows that the performance of GASTL is not sensitive to the value of m. The results are listed in Table 4.3.3, where we can find that the performance of GASTL is quite stable with respect to the balance parameters λ and γ for Caltech101, IMDB, and Twitter. For ESC-50, the standard deviation values are relatively large compared with the mean values.

4.3.4 Performance Comparison

We present the classification accuracy results of GASTL and baselines on all datasets in Tables 4.3 to 4.7, corresponding to Caltech101 (SIFT-BOW), Caltech101 (VGG-19), IMDB, Twitter, and ESC-50[16], respectively. Note that "No Transfer" denotes the method that performs classification on target samples without knowledge transfer. In Tables 4.3, 4.4, and 4.7 each column corresponds to one subset, while in

[16]Due to limited space, for ESC-50 we only perform results with 10 training samples for each target class.

Set ID / Method	1	2	3	4	5
No Transfer	42.33	61.67	42.33	48.33	46.00
STL	46.00	64.33	48.33	46.33	56.00
RDSTL	36.67	51.33	37.00	39.67	42.00
S-Low	35.00	51.00	36.67	34.33	36.67
KMM-Soft	48.67	66.00	47.33	52.67	56.33
KMM-Hard	43.67	63.67	43.33	47.67	48.33
MLS-Soft	47.33	66.33	47.67	51.67	53.33
MLS-Hard	45.33	62.00	42.00	46.00	47.67
GASTL-SoftA	**53.00**	**67.67**	**51.67**	**56.00**	**58.67**
GASTL-HardA	47.67	64.33	47.67	48.67	52.33
GASTL-SoftB	**52.33**	**68.00**	**51.33**	**53.33**	**58.67**
GASTL-HardB	48.00	64.00	46.33	49.33	51.00

Table 4.3. Performance of GASTL and competing feature selection algorithms in classification on Caltech101 with SIFTBOW as feature. Classification accuracy (%) is used as the evaluation metric.

Tables 4.5 and 4.6, each column corresponds to one training sample number in each target class. We highlight the best two performances in each experiment given that we find in many cases the best two (or even more) performance are very close to each other.

We first provide an overall description on the comparison between GASTL and the competitors on each dataset. We can find that in Tables 4.3 and 4.4 the best performances are claimed by GASTL methods. In Tables 4.5 and 4.6, RDSTL is comparable to GASTL in a few cases, while in Table 4.7, the advantages of GASTL over RDSTL are sometimes small. In other words, GASTL provides the best overall performance. We can also find that the classification performance of the two pseudo-labeling schemes are quite similar to each other in almost every case in the five tables. Our next analysis focuses on the comparison between performance generated by soft classifiers and hard classifiers.

In Table 4.3, it is obvious that GASTL methods with soft classifiers provide the best performance. For image datasets such as Caltech101, it is unusual for the rel-

Method \ Set ID	1	2	3	4	5
No Transfer	94.33	95.67	93.00	94.33	93.67
STL	95.00	95.33	92.00	94.67	94.33
RDSTL	91.67	93.00	85.67	90.00	89.33
S-Low	91.33	90.67	87.00	89.67	89.33
KMM-Soft	95.33	96.00	93.33	93.67	94.67
KMM-Hard	94.33	95.33	92.33	94.00	93.67
MLS-Soft	94.00	95.33	92.67	94.00	93.67
MLS-Hard	95.00	95.67	93.00	94.33	93.67
GASTL-SoftA	95.67	**97.00**	93.67	95.00	**96.00**
GASTL-HardA	95.00	**97.00**	**94.67**	**95.67**	**96.00**
GASTL-SoftB	**96.00**	**97.00**	93.67	94.67	**96.00**
GASTL-HardB	**96.00**	96.67	**94.67**	**95.67**	95.67

Table 4.4. Performance of GASTL and competing feature selection algorithms in classification on Caltech101 with VGG19 as feature. Classification accuracy (%) is used as the evaluation metric.

evance between one source sample and a particular target class to be much larger than for other target classes. A soft classifier is able to characterize the relationship between a source sample and each target class during pseudo-labeling, while a hard classifier only selects the most similar class to each source sample and ignores other target classes, which may degrade knowledge transfer performance due to the possible useful information from other classes. This can also be validated by the results of KMM and MLS in Table 4.3, which shows the advantages of soft classifiers over hard classifiers. However, the classification rates listed in Table 4.4 are quite large and similar to each other. Therefore, these results cannot provide significant information on validating the advantages of soft classifier over hard classifier on image datasets. On the other hand, the performance increase brought by knowledge transfer also depends on the difficulty of the classification problem. For example, the performance increase of Caltech101 using the SIFT-BOW features is much larger than using the VGG-19 features.

Method \ TS	10	100	1000
No Transfer	57.60	68.20	73.27
STL	58.80	69.53	73.73
RDSTL	**60.53**	**70.47**	73.47
S-Low	59.93	64.87	73.07
KMM-Soft	57.13	69.13	73.53
KMM-Hard	58.00	68.60	73.33
MLS-Soft	56.40	68.47	72.87
MLS-Hard	56.80	68.80	72.53
GASTL-SoftA	59.80	69.33	74.60
GASTL-HardA	**60.73**	**70.47**	**77.67**
GASTL-SoftB	60.33	69.27	74.47
GASTL-HardB	**60.53**	**70.60**	**77.67**

Table 4.5. Performance of GASTL and competing feature selection algorithms in classification on IMDB. Classification accuracy (%) is used as the evaluation metric. TS = Training sample number in each target class.

In Tables 4.5 and 4.6, hard classifiers consistently provide slightly better performance than soft classifiers for GASTL methods, while for KMM and MLS, the differences are smaller. According to our experimental setup, IMDB and Twitter play interchangable roles as source and target. Therefore, in each experiment both source and target domains share a label space with two labels ("positive sentiment" and "negative sentiment"). Therefore, in this case it is better to use hard classifier than soft classifier since the two labels indicate two multually exclusive categories.

It is difficult to interpret the results in Table 4.7. The overall performance of hard classifier based methods is better than soft classifier based ones for groups 1 to 4, while group 5 provides opposite results. We believe that the results reflect the complexity of this dataset.

Method \ TS	10	100	1000
No Transfer	55.47	58.93	77.80
STL	53.67	58.93	76.93
RDSTL	**58.60**	59.93	63.13
S-Low	55.87	59.47	64.47
KMM-Soft	56.27	61.20	78.27
KMM-Hard	57.47	60.33	78.47
MLS-Soft	56.27	59.40	77.93
MLS-Hard	57.27	60.40	78.13
GASTL-SoftA	56.67	61.93	78.20
GASTL-HardA	57.87	**62.73**	**78.53**
GASTL-SoftB	56.67	**62.00**	78.20
GASTL-HardB	**58.67**	61.93	**78.60**

Table 4.6. Performance of GASTL and competing feature selection algorithms in classification on Twitter. Classification accuracy (%) is used as the evaluation metric. TS = Training sample number in each target class.

4.3.5 Discussion

4.3.5.1 The Effect of Local Data Structure Preservation

As mentioned in Chapter 4.2.1.1, local data structure preservation provides similar representation for nearby data points. Intuitively, local data structure preservation applied in the hidden layer of the autoencoder is likely to improve knowledge transfer performance because it is able to reduce hidden layer representation distortion as they are involved in data reconstruction and pseudo-labeling. In order to measure the effect of local data structure preservation on knowledge transfer, we compare the classification performance when $\gamma = 0$ with the optimal one. In Fig. 4.2, the comparions on one set of Caltech101 with both SIFTBOW and VGG19 as the input to autoencoder, both IMDB and Twitter with each target data having 10 training samples, and one set of ESC-50 are displayed. We can find that in most cases setting $\gamma = 0$ cannot achieve the optimal performance. Exceptions appear in the cases of "HardA" on Caltech101 with SIFTBOW features and both "HardA" and "HardB" on Twitter. Due to our observations on complete comparisons, classification rates

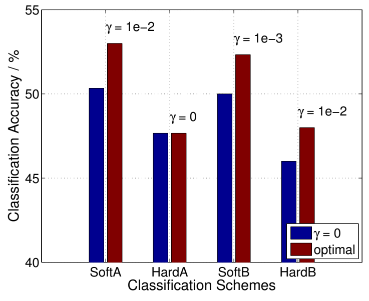
Method \ Set ID	1	2	3	4	5
No Transfer	20.50	25.00	20.00	19.00	20.00
STL	19.00	19.00	16.00	18.00	20.50
RDSTL	**26.00**	28.00	**25.50**	**26.50**	**26.50**
S-Low	17.00	15.50	16.00	17.50	15.00
KMM-Soft	21.00	28.00	20.00	22.00	25.00
KMM-Hard	24.00	28.50	23.00	22.00	20.00
MLS-Soft	22.50	26.00	21.00	19.50	25.00
MLS-Hard	23.00	27.50	22.50	22.50	22.00
GASTL-SoftA	**26.00**	27.50	23.00	23.00	26.00
GASTL-HardA	**26.00**	**30.50**	23.50	24.50	24.50
GASTL-SoftB	**26.50**	28.00	23.50	23.50	**26.50**
GASTL-HardB	**26.00**	**29.50**	**24.50**	**25.00**	24.50

Table 4.7. Performance of GASTL and competing feature selection algorithms in classification on ESC-50. Classification accuracy (%) is used as the evaluation metric.

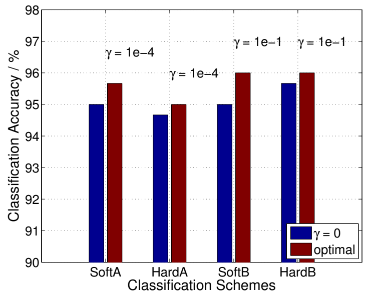
resulted from the situation of $\gamma = 0$ are consistently lower than those resulted from the situations of $\gamma \neq 0$. Therefore, the former exception can be regarded as a subtle outlier. However, we found that the advantages in classification accuracy contributed by local data structure preservation are not obvious on Twitter. One possible explanation is that the local data structures in the space of WORD2VEC feature cannot provide discriminative information for samples in Twitter dataset. Therefore, local data structure preservation negatively affected the knowledge transfer performance refected by classification accuracy on unlabeled target samples.

4.3.5.2 The Effect of Source Sample Selection

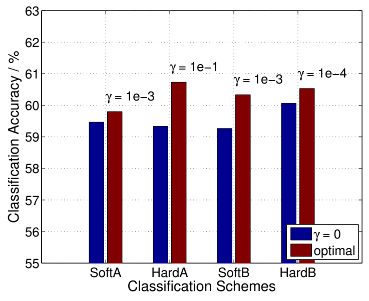
We claim that transferring knowledge from source samples indiscriminately may cause negative transfer since there is no guarantee that all source samples have sufficient relevance with target domain. In order to demonstrate the advantages of source sample selection, we compare the classification accuracy for three different cases: the optimal value of p found with GASTL, $p = 0$ (i.e., no transfer learning), and $p = n_{src}$ (i.e., no sample selection). The results shown in Fig. 3 demonstrate not only that



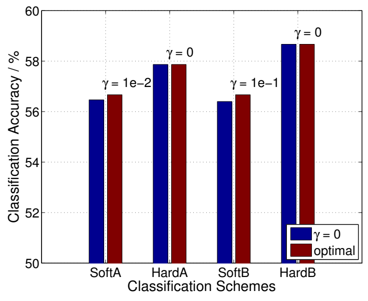
(a) Caltech101 (SIFTBOW)



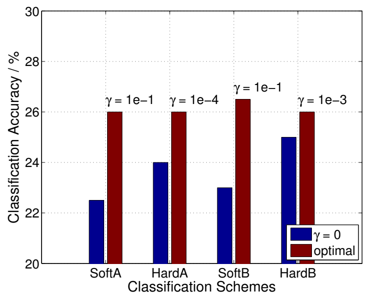
(a) Caltech101 (VGG19)



(b) IMDB

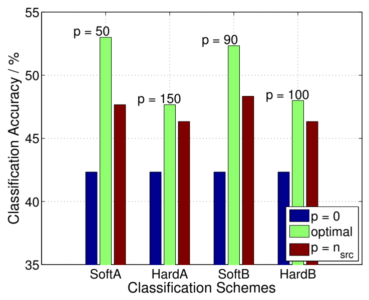


(c) Twitter

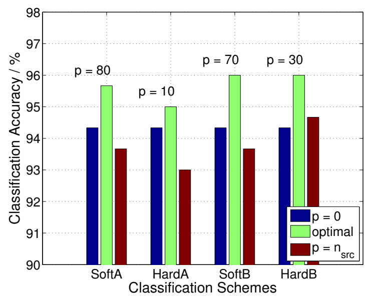


(d) ESC-50

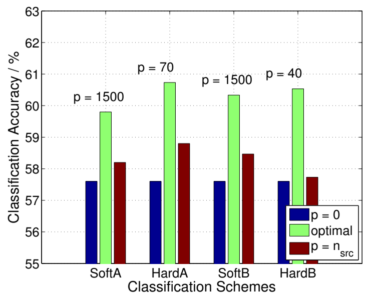
Figure 4.2. Comparison of GASTL performance when $\gamma = 0$ and optimal GASTL performance. Classification accuracy (%) is used as the evaluation metric. Optimal values for γ are shown.



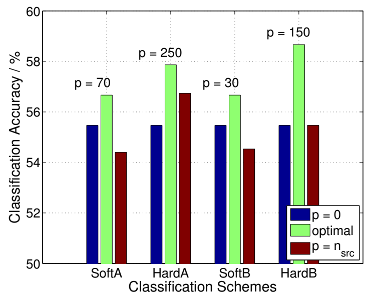
(a) Caltech101 (SIFTBOW)



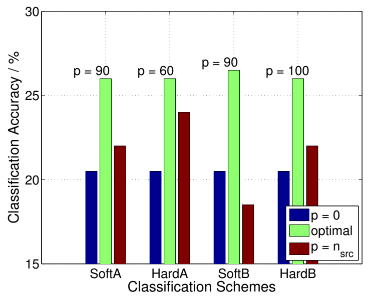
(a) Caltech101 (VGG19)



(b) IMDB



(c) Twitter



(d) ESC-50

Figure 4.3. Comparison of GASTL performance when all source samples are used for classifier training and optimal GASTL performance. Classification accuracy (%) is used as the evaluation metric. Optimal values for p are shown.

significant performance gains are obtained via GASTL, but also that in many cases the blind consideration of all source samples can in fact result in negative transfer, as seen by the reduced performance obtained with $p = n_{src}$ versus $p = 0$.

CHAPTER 5
FEW-SHOT LEARNING

Few-shot learning is a technique that feeds a learning model with a very small amount of labeled training data. Compared with traditional machine learning methods which use large amounts of training data, few-shot learing methods can significantly save the time and labor cost in training data collection. In this chapter, we propose a few-shot learning method for wearable sensor based human activity recognition, which can be treated as a type of pattern recognition problem that seeks high-level human activity knowledge from low-level sensor inputs. Due to the high costs to obtain human generated activity data and the ubiquitous similarities between activity modes, it would be more economical to borrow information from existing activity data than to collect more data and train a new model for new activities with only a few data at hand. To be more specific, in this chapter we consider knowledge transfer between domains which do not have identical labels.

5.1 Introduction

Studies in human activity recognition (HAR) have been attracting increasing attentions in recent years due to their potential applications in human health care [107], indoor localization [108], smart hospital [109], and smart home [58]. HAR is a technique that aims at predicting people's activities from the low-level sensor inputs. In recent years, the application of wearable devices such as smart wristbands and smart phones in HAR significantly facilitates HAR research due to their small sizes and low power consumption [110], as well as the capability of real-time information capturing [111].

Traditional machine learning (ML) methods such as support vector machine and decision trees had significantly promoted the development of HAR studies during the past few decades. However, limitations such as heavy reliance on human domain knowledge [112] and shallow feature extraction impede these ML methods to provide satisfying results in most daily HAR tasks [62]. During recent years, deep learning methods have been continuously providing marvelous peroformance in many areas such as computer vision and natural language processing [113]. The capability to automatically extract high-level features makes deep learning methods largely alleviated from the drawbacks of conventional ML methods. There have been many deep learning based HAR works proposed in recent years [71, 73, 75, 114]. However, the training time and the amount of data required for deep learning systems are always much larger than that of traditional ML systems. Furthermore, the large time and labor costs makes it difficult to build a large-scale labeled dataset with high quality. Therefore, it is necessary to find a way to alleviate these problems of deep learning.

Few-shot learning (FSL) [55] is a type of transfer learning technique [43] aiming at learning a classifier to recognize unseen classes (*target* domain) with only a small amount of labeled training samples by using knowledge from existing models on relevant classes (*source* domain). In this paper, we propose a novel deep FSL method for unseen activity recognition. We first transfer structures and parameters of front layers (feature extractor) from a trained neural network to make use of the general knowledge among similar tasks they contain, then we calculate similarities between source classes and target classes and train a target classifier. Our contribution is two-fold:

- We design a deep neural network for HAR.

- We propose a few-shot learning framework to transfer knowledge from exisiting classes to new classes.

To the best of our knowledge, this is the first work of deep learning based few-shot learning on HAR.

5.2 Proposed Method

In this section, we introduce our proposed cross-domain human activity recognition (CDHAR) approach. We follow the standard deep learning based transfer learning procedure for our CDHAR model. Below is a framework for our proposed CDHAR method. We first train a deep neural network with source data.

5.2.1 Basic Framework

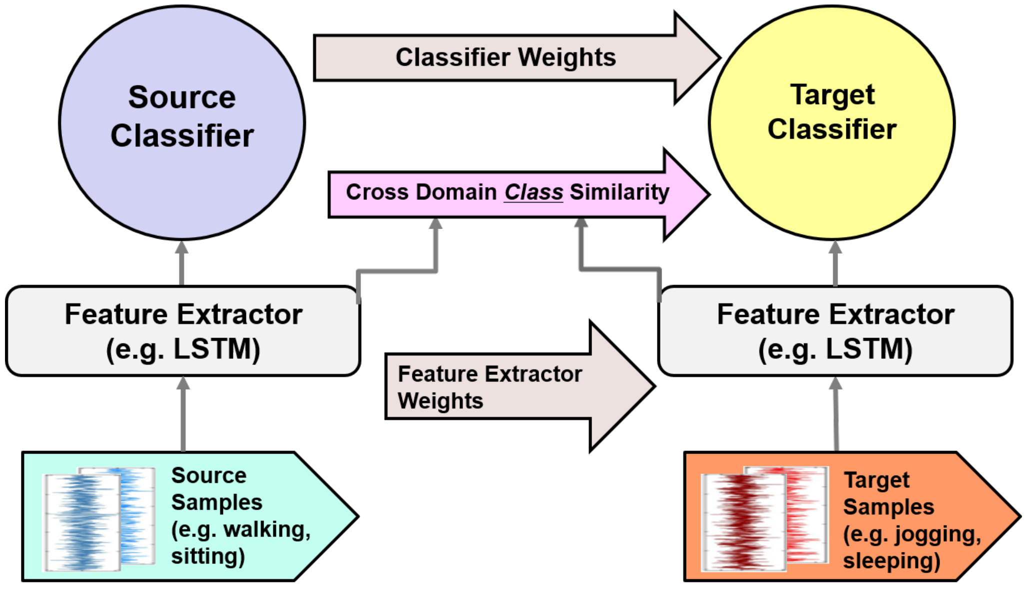

Source Classifier
Classifier Weights
Target Classifier
Cross Domain *Class* Similarity
Feature Extractor (e.g. LSTM)
Feature Extractor (e.g. LSTM)
Feature Extractor Weights
Source Samples (e.g. walking, sitting)
Target Samples (e.g. jogging, sleeping)

Figure 5.1. Basic Framework

The basic framework for CDHAR is illustrated in Fig. 5.1. We first train a source network with source domain samples to get the source feature extractor $f(\theta_{\text{src}})$, with parameters θ_{src}, and the source classifier $C(\cdot|\mathbf{W}_{\text{src}})$, with parameters $\mathbf{W}_{\text{src}} \in \mathbb{R}^{c_{\text{src}} \times d}$, where c_{src} is the number of source classes and d is the dimensionality of encoded features fed into the source classifier. We empirically use a stacked LSTM with two hidden layers as feature extractor and two fully connected layers followed by a softmax predictor as classifier for both source and target networks. We use LSTM as feature extractor with the purpose of taking advantages of the temporal dependencies within the HAR data, as the layout of an LSTM layer forms a directed cycle, where the states of the network in current timestep depends on those of the network in the previous timestep. The network structure for both the source domain and the target domain is shown in Fig. 5.2.

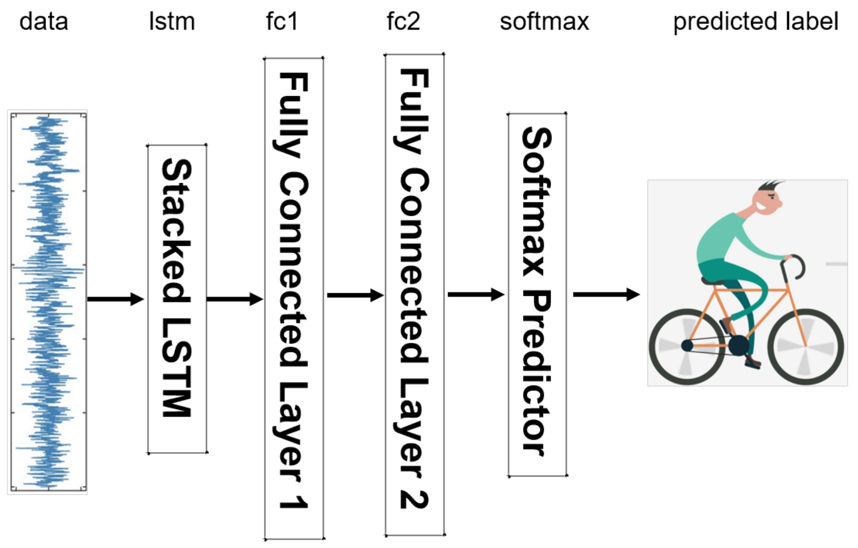


Figure 5.2. Network Structure

For a deep neural network, features in the lower layers are generic while features in upper layers are more specific to tasks. For transfer learning purposes, we use the source feature extractor parameters θ_{src} as the initialization of the target feature extractor parameters. With generic features being transferred, the next step is to transfer information in source classifier to target classifier. However, since information in a classifier is highly specific to the task, it is necessary to pick information relevant to the target task from the source classifier in order to avoid negative transfer. In our problem scenario, we focus on the class-wise similarities between the source and target domain samples. In this paper, we use the magnitudes of the reconstruction coefficients to measure cross-domain class-wise similarity. We assume a linear mapping between source domain samples and target domain samples, that is, $f_1(\mathbf{X}_{\text{trg}}) = \mathbf{A}^T f_2(\mathbf{X}_{\text{src}})$, where $\mathbf{A} \in \mathbb{R}^{n_{\text{src}} \times n_{\text{trg}}}$ is a reconstruction matrix with element values indicating cross-domain sample-wise similarity, and $f_1(\cdot)$ and $f_2(\cdot)$ are two appropriate networks to embed $\mathbf{X}_{\text{trg}}$ and $\mathbf{X}_{\text{src}}$, respectively. In our problem scenario, the functions $f_1(\cdot) = f_2(\cdot)$ are the source feature extractor. We first solve the following minimization problem to get a reconstruction matrix $\mathbf{A}$:

$$\min_{\mathbf{A}} \frac{1}{2n_{\text{trg}}} \|\mathbf{A}^T f(\mathbf{X}_{\text{src}}) - f(\mathbf{X}_{\text{trg}})\|_F^2 + \lambda \|\mathbf{A}\|_{2,1}, \tag{5.1}$$

where λ is a balance parameter and $f(\cdot)$ denotes the source feature extractor. Since each row of $\mathbf{A}$ indicates the importance of the corresponding encoded source sample in reconstrucing encoded target samples, we use the ℓ_2-norm of each row of $\mathbf{A}$ to measure the relevance between an encoded source sample and encoded target samples, which leads to the $\ell_{2,1}$-norm regularization term in (5.1) that enforces row sparsity on the transformation matrix $\mathbf{A}$ for similarity measure. Since transferring knowledge from source samples that are inversely correlated with target samples may cause negative transfer, we multiply each element in $\mathbf{A}$ with the rectified linear unit (ReLU) activation function so that $\mathbf{A}$ only contains similarity measure between positively correlated source and target domain samples. With the reconstruction matrix $\mathbf{A}$, we sum up the element values within each source-target class pair to get a class-wise similarity matrix $\mathbf{O} \in \mathbb{R}^{n_{\text{csrc}} \times n_{\text{ctrg}}}$. That is,

$$\mathbf{O}^{(p,q)} = \sum_{i \in \text{ class } p} \sum_{j \in \text{ class } q} \mathbf{A}^{(i,j)} \tag{5.2}$$

We then propose two schemes to get a normalized class-wise similarity matrix $\mathbf{W} \in \mathbb{R}^{n_{\text{csrc}} \times n_{\text{ctrg}}}$.

- **Scheme A**: Soft normalization

$$\mathbf{W}^{(p,q)} = \frac{\mathbf{O}^{(p,q)}}{\sum_{p=1}^{n_{\text{csrc}}} \mathbf{O}^{(p,q)}} \tag{5.3}$$

- **Scheme B**: Hard normalization

$$\mathbf{W}^{(p,q)} = \begin{cases} 1 & \mathbf{O}^{(p,q)} = \{\max_i \mathbf{O}^{(i,q)}\}_{i=1}^{n_{\text{csrc}}} \\ 0 & \text{Otherwise} \end{cases} \tag{5.4}$$

The initialization of target classifier weights is a linear combination of the trained source classifier weights based on the normalized class-wise similarity matrix $\mathbf{W}$, which is $\mathbf{W}_{\text{trg}} = \mathbf{W}^T \mathbf{W}_{\text{src}}$. Compared with hard normalization, soft normalization may help improve knowledge transfer performance since it is able to capture the relationship between each single target class and multiple source classes instead of one. This is important for HAR tasks since there sometimes exists commonalities between activity categories.

Both θ_{src} and $\mathbf{W}_{\text{trg}}$ are used as initialization for feature extractor and classifier parameters in target network. Given the pre-trained network, the final target feature extractor parameter θ_{trg}^* and classifier parameter $\mathbf{W}_{\text{trg}}^*$ are obtained through fine-tuning.

5.2.2 Implementation

We used PyTorch [115] to implement the described framework. Network parameter optimization was performed via Adam [116]. We employ the method proposed in

[117][1] to solve the optimization problem (5.1) , which efficiently solves an an $\ell_{2,1}$-norm optimization problem by reformulating it as two equivalent smooth convex optimization problem.

5.3 Experiments

In this section, we evaluate the knowledge transfer performance of CDHAR. Experiments are conducted on two benchmark human activity datasets. We also compare CDHAR with other relevant state-of-the-art transfer learning techniques. As mentioned in Section 5.2.1, we first train a neural network solving a source domain classification task. After that, we build a network for target domain with the same structure as that for the source domain. We use the feature extractor weights from the source network as the initialization of feature extractor weights and a weighted linear combination of source classifier based on a class-wise similarity measure between source and target domains as the initialization of classifier weights to train a target neural network. The classification rates on target testing samples are used as metric to evaluate knowledge transfer performance.

5.3.1 Dataset Information

We first provide the overall information of each dataset and introduce the source/target domain setup. We perform experiments on two benchmark datasets: the the opportunity dataset (OPP) [118] and the PAMAP2 dataset (PAMAP2) [119]. The opportunity dataset (OPP) [118] consists of common kitchen activities from 4 participants with wearable sensors. Data from 5 different runs are recorded for each participant with activities being annotated with 18 mid-level gesture annotations. Following [114], we only keep data from sensors without any packet-loss, which includes ac-

[1]Codes available at: `https://github.com/jundongl/scikit-feature/blob/master/skfeature/function/sparse_learning_based/ls_l21.py`

celerometer data from the upper limbs and the back, and complete IMU data from both feet, so that the resulting dataset has 77 dimensions. The PAMAP2 dataset (PAMAP2) [119] consists of 12 household and exercise activities from 9 participants with wearable sensors. The dataset has 53 dimensions. In order to eliminate the side effects caused by imbalanced classes, we set the number of samples from each class to be the same within each dataset through random selection. We keep 202 samples and 129 samples for OPP and PAMAP2, respectively. Following [114] for frame-by-frame analysis, a sliding window with a one-second duration and 50% overlap is performed and the resulting data are used as inputs to the system for both datasets. After preprocessing, each sample is represented in the form of a matrix.

5.3.2 Source/Target Split

We test the knowledge transfer performance in two scenarios, i.e. transferring knowledge from other activities on same people and transferring knowledge across both activities and individuals. In both scenarios, we split the activities into two groups on each dataset for knowledge transfer purpose. Details on the source/target split for the different activities are listed in Table 5.1. For knowledge transfer across both activities and individuals, the individuals are partitioned into different groups for the source/target split. In OPP, the target domain includes target activities of one individual and the source domain includes source activities of the remaining individuals. In PAMAP2, individuals are first partitioned into three groups. The target domain includes target activities of one group and the source domain includes source activities of the remaining groups. We repeat experiments until all individuals/groups have the chance to be employed as both source and target domain in each dataset.

Split / Dataset	Source Activities	Target Activities
OPP	Open Door 2 Close Door 2 Close Fridge Close Dishwasher Close Drawer 1 Close Drawer 2 Close Drawer 3 Clean Table Drink from Cup Toggle Switch	Open Door 1 Close Door 1 Open Fridge Open Dishwasher Open Drawer 1 Open Drawer 2 Open Drawer 3
PAMAP2	Lying Standing Walking Running Ascending Stairs Vacuum Cleaning Rope Jumping	Sitting Cycling Nordic Walking Descending Stairs Ironing

Table 5.1. Source/target split for activities

5.3.3 Baselines

We compare CDHAR with the following five baselines. Note that all baselines are performed with the network structure described in Section 5.2.1. The parameters for neural networks are listed in Table 5.2.

- *No Transfer*, which trains the designed network with target training data from scratch, without any knowledge transfer from source network.
- *Half Transfer*, which only transfers the source feature extractor weights as initialization for target feature extractor weights. The target classifier is trained from scratch using the target training data.
- *CDHAR-NGD*, which uses normalized Google distance (NGD) to measure cross-domain class-wise similarity in a semantic way [120], with other steps the same as CDHAR.

Parameters	OPP	PAMAP2
LSTM Layer	2	2
LSTM Hidden Size	64	50
FC1 Size	64	50
FC2 Size	64	25

Table 5.2. Network Structure for Both Datasets

- *CDHAR-Cls*, which first trains networks for both source and target domain from scratch and then calculates the cosine value between classifier weights of each source-target-class-pair as the cross-domain class-wise similarity measure [121], with other steps the same as CDHAR.
- *CDHAR-Corr*, which uses cosine similarity as cross-domain *sample-wise* similarity measure [56], with other steps the same as CDHAR.

5.3.4 Performance Comparison

We present the classification accuracy results of CDHAR and baselines on both datasets in Fig. 5.3-5.6, including different combinations of datasets (OPP or PAMAP2), whether source data and target data being generated from the same participant/groups of participants or not, and number of training samples from each target class (1 or 5). Each number is an average of results from 100 repetitions with varying target training data. Analysis on these results is provided as follows.

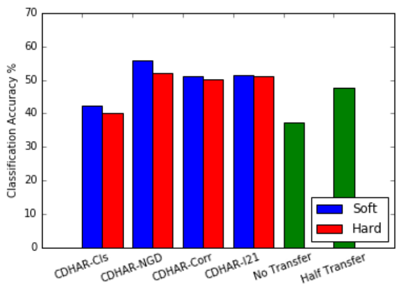
- *Transfer Learning and Negative Transfer*: We can find that most transfer learning based methods perform bettern than "No Transfer" and "Half Transfer", except for "CDHAR-Cls". However, we also see negative transfer when only the source feature extractor weights are transferred. We conjecture that the possible reason is that the weights we transferred are generated from all source domain samples without any regard for similarity measure and selection. Therefore source samples with different relevance with target domain make equal contributions to the weights and those with weak relevance may provide harmful information during training. The consistent poor performance of "CDHAR-Cls" also results from negative transfer. The severe insufficiency of target training samples make the trained target classifier suffer significantly from overfitting. Therefore, the similarity between the weights of the target classifier and the source classifier is very likely to deviate from the true cross-domain class-wise similarity.

- *Number of Training Samples*: Increasing the number of training samples for each target class from 1 to 5 can bring an increase about 10% to 15% in performance for each method. It is also obvious that gap between "No Transfer", baseline without knowledge transfer and other transfer learning methods shrinks. We believe that by as the number of target training samples increase, the gap between performance of "No Transfer" and transfer learning based methods will keep shrinking, which decreases the necessity of transfer learning.

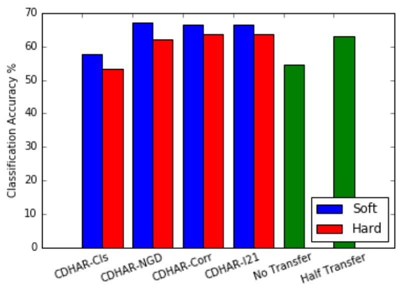
- *Source Data*: For OPP, using source data from the same participant consistently provides better performance than that of using source data from different participants. Our method assumes that data from the same participants may have similar marginal distribution, though they have different conditional distribution due to disjoint label space between source domain and target domain. However, this is not true for PAMAP2. We conjecture that the combination of 3 participants into a group may introduce incompatibility in marginal distribution, which may leads to negative transfer.

- *Soft Normalization vs. Hard Normalization*: It is obvious that soft normalization does better than hard normalization in most cases. For activity data, each activity may be related with multiple other activities instead of only a specific one another. Soft normalization is able to characterize the relationship between a target activity with source activities during similarity measure, while hard normalization only selects the most similar source activity to each target activity, which may degrade knowledge transfer performance due to the possible useful information from other source activities.

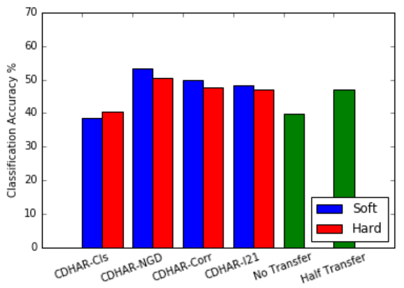
- *Cross-Domain Class-Wise Similarity Measures*: For OPP, we find that "CDHAR-l21" and "CDHAR-Corr" provides similar performance in almost all cases. When each target class provides only one sample for training, "CDHAR-NGD" does better than both "CDHAR-l21" and "CDHAR-Corr" while when five samples are used, these three methods performs similarly. Due to the source/target split scheme we use for OPP, it is easy for NGD to find similarities between source and target classes using the semantic information in their labels. This also explains the fact that the advantages of "CDHAR-NGD" is not significant for PAMAP2.



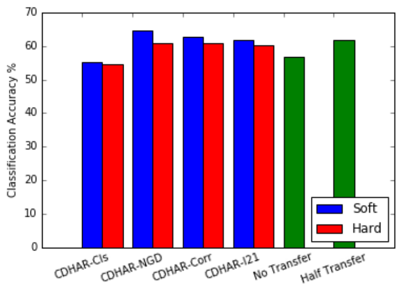
(a) Participant 1, one-shot



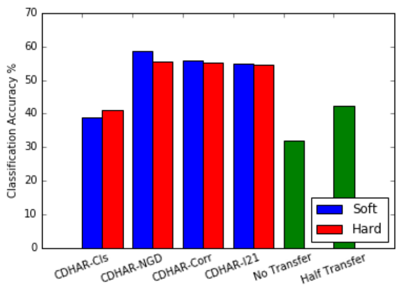
(b) Participant 1, five-shot



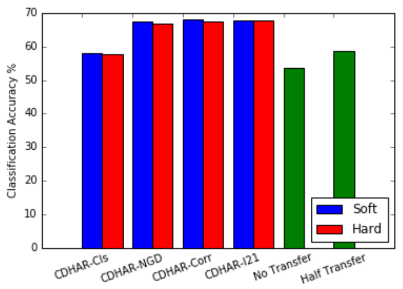
(c) Participant 2, one-shot



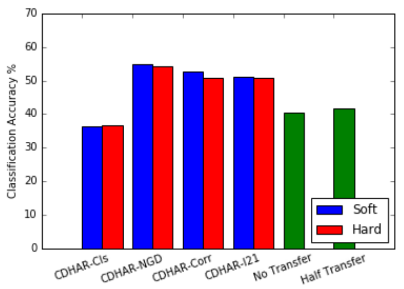
(d) Participant 2, five-shot



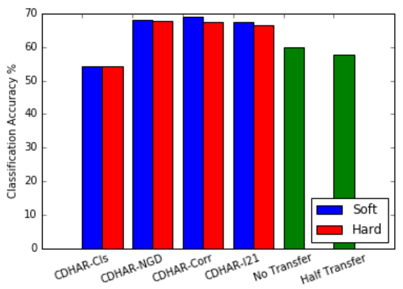
(e) Participant 3, one-shot



(f) Participant 3, five-shot

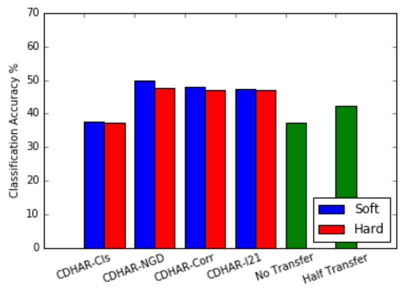


(g) Participant 4, one-shot

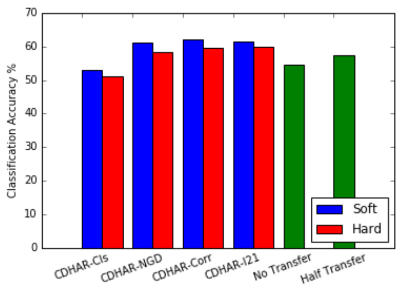


(h) Participant 4, five-shot

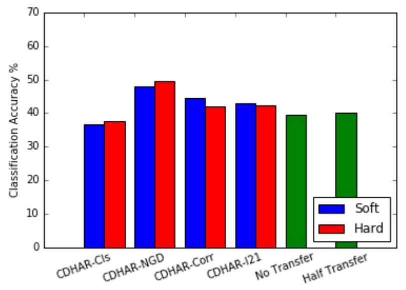
Figure 5.3. Comparison of CDHAR performance with OPP dataset. Source data and target data are generated from the same participant. Subfigures in the left column show results when each target class provides one training sample for target network training. Subfigures in the right column shows results when each target class provides five training samples for target network training. Classification accuracy (%) is used as the evaluation metric.



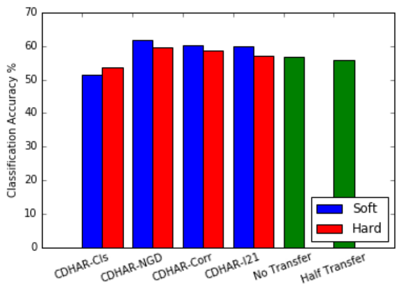
(a) Participant 1, one-shot



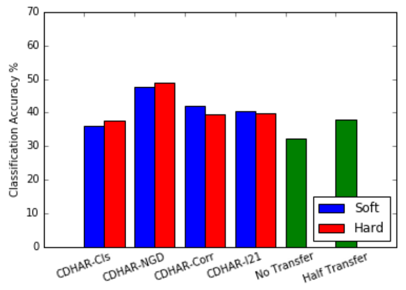
(b) Participant 1, five-shot



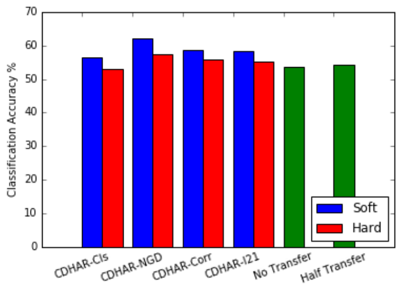
(c) Participant 2, one-shot



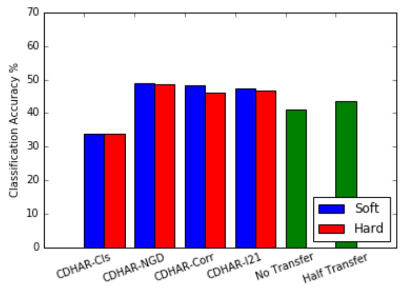
(d) Participant 2, five-shot



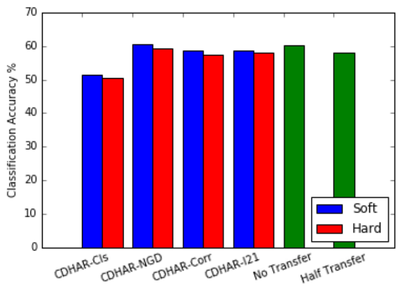
(e) Participant 3, one-shot



(f) Participant 3, five-shot

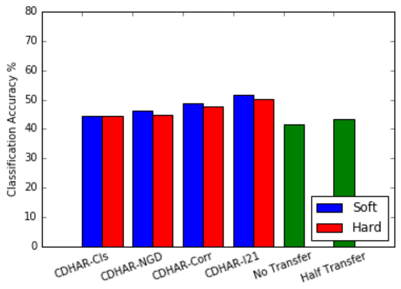


(g) Participant 4, one-shot

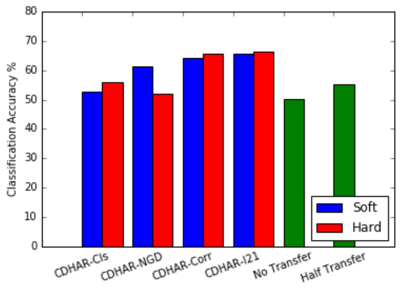


(h) Participant 4, five-shot

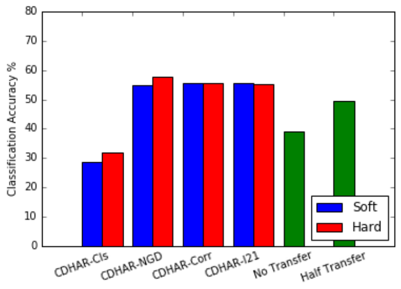
Figure 5.4. Comparison of CDHAR performance with OPP dataset. Source data and target data are generated from the different participants. Subfigures in the left column show results when each target class provides one training sample for target network training. Subfigures in the right column shows results when each target class provides five training samples for target network training. Classification accuracy (%) is used as the evaluation metric.



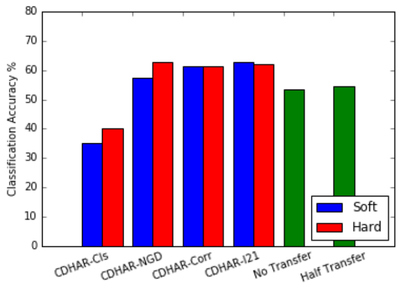
(a) Group 1, one-shot



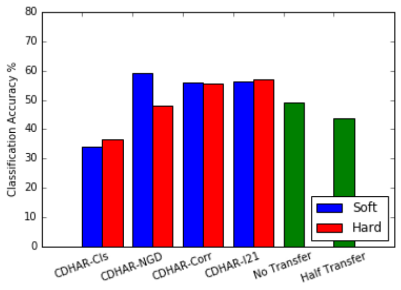
(a) Group 1, five-shot



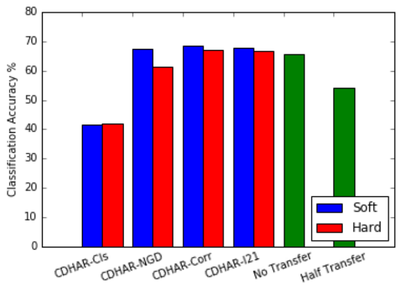
(a) Group 2, one-shot



(a) Group 2, five-shot

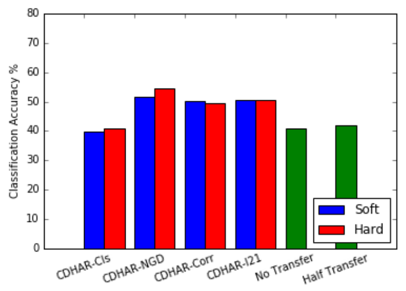


(b) Group 3, one-shot

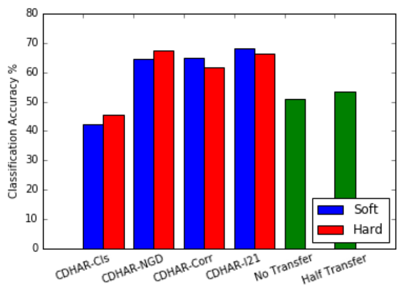


(b) Group 3, five-shot

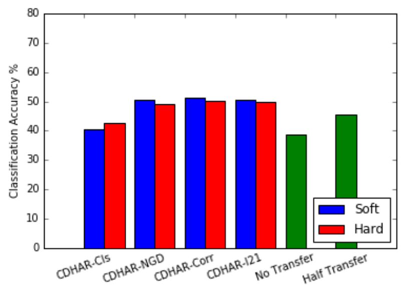
Figure 5.5. Comparison of CDHAR performance with PAMAP2 dataset. Source data and target data are generated from people from the same group. Subfigures in the left column show results when each target class provides one training sample for target network training. Subfigures in the right column shows results when each target class provides five training samples for target network training. Classification accuracy (%) is used as the evaluation metric.



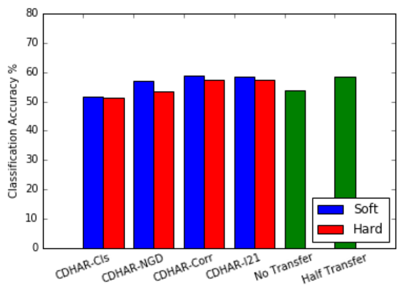
(a) Group 1, one-shot



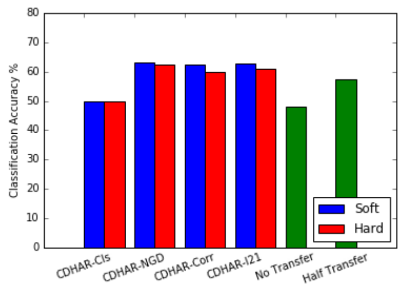
(a) Group 1, five-shot



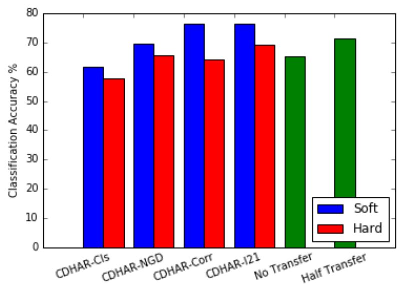
(a) Group 2, one-shot



(a) Group 2, five-shot



(b) Group 3, one-shot



(b) Group 3, five-shot

Figure 5.6. Comparison of CDHAR performance with PAMAP2 dataset. Source data and target data are generated from people from the different groups. Subfigures in the left column show results when each target class provides one training sample for target network training. Subfigures in the right column shows results when each target class provides five training samples for target network training. Classification accuracy (%) is used as the evaluation metric.

CHAPTER 6

CONCLUSION AND FUTURE WORK

6.1 Conclusion

In this thesis, we have applied information selection in three different machine learning problems.

In Chapter 3, we proposed a graph and autoencoder-based unsupervised feature selection (GAFS) method. Unlike similar existing techniques that combine sparse learning and feature selection, the proposed method projects the data to a lower-dimensional space using a single-layer autoencoder, in contrast to the linear transformation used by most existing methods. With our proposed framework, we bypass the limitation of existing methods with linear dimensionality reduction schemes, which may lead to performance degradation for datasets with richer structure that is predominant in modern datasets. Experimental results demonstrate the advantages of GAFS versus methods in the literature for both classification and clustering tasks.

In Chapter 4, we propose a graph and autoencoder based self-taught learning (GASTL) method. The main innovations in our self-taught learning methodology with respect to the literature can be summarized as (a) leveraging relevance metrics to select a subset of source samples in transfer learning; (b) considering cross domain relevance for classifier training; and (c) developing our method for hard as well as soft classification problems. With our proposed framework, we decrease negative transfer and improve knowledge transfer performance in many scenarios. Experimental results demonstrate the advantages of GASTL versus methods in the literature.

In Chapter 5, we propose a few-shot learning framework for human activity recognition. The main contributions in this proposed approach is three-fold. First, we propose a way to transfer feature extractor and classifier from the source network to the target network based on cross-domain class relevance. Second, the proposed framework is a general framework where different cross-domain class relevance measure can be embedded. With the proposed framework, satisfying human activity recognition results can be achieved even when only very few training samples are available for each class. Experimental results show the advantages of the framework over methods with no knowledge transfer or that only transfer knowledge of feature extractor.

6.2 Future Work

There are still many problems that are worth exploring regarding the three applications of information selection proposed in our thesis.

For unsupervised feature selection, the work we present here is our first attempt to leverage autoencoders for unsupervised feature selection purposes. Therefore, we use the most standard setting for the construction of the autoencoder, e.g., there is no desired or particular structure to the activations or the reconstruction error. In the future, we plan to explore the effectiveness of more elaborate versions of an autoencoder for feature selection purposes. Furthermore, by employing label information, we can also extend our work to a supervised feature selection framework.

For self-taught learning, our work can be easily extended from a single-layer autoencoder-based design to one based on deep neural networks. We also plan to integrate discriminative information of target samples into our framework.

For few-shot learning, we may combine statistical ways and semantic ways to measure cross-domain class-wise similarity instead of separating them. It is also worth trying to find a way to combine source data from both same and different participants from the target participant to see if knowledge transfer performance can be improved.

BIBLIOGRAPHY

[1] M. Pal and G. M. Foody, "Feature selection for classification of hyperspectral data by SVM," *IEEE Trans. Geosci. Remote Sens.*, vol. 48, no. 5, pp. 2297–2307, 2010.

[2] H. Liu, X. Wu, and S. Zhang, "Feature selection using hierarchical feature clustering," in *CIKM*, 2011, pp. 979–984.

[3] L. O. Jimenez and D. A. Landgrebe, "Supervised Classification in High-Dimensional Space: Geometrical, Statistical, and Asymptotical Properties of Multivariate Data," *IEEE Trans. Syst. Man Cybern. C Appl. Rev.*, vol. 28, no. 1, pp. 39–54, 1998.

[4] X. Lu, Y. Wang, and Y. Yuan, "Sparse coding from a Bayesian perspective," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 24, no. 6, pp. 929–939, 2013.

[5] X. Zhu, S. Zhang, Z. Jin, Z. Zhang, and Z. Xu, "Missing value estimation for mixed-attribute data sets," *IEEE Trans. Knowl. Data Eng.*, vol. 23, no. 1, pp. 110–121, Jan. 2011.

[6] Y. Yang, Z. Ma, F. Nie, X. Chang, and A. G. Hauptmann, "Multi-class active learning by uncertainty sampling with diversity maximization," *Int. J. Comput. Vis.*, vol. 113, no. 2, pp. 113–127, 2015.

[7] L. Song, A. Smola, A. Gretton, K. M. Borgwardt, and J. Bedo, "Supervised feature selection via dependence estimation," in *ICML*, 2007, pp. 823–830.

[8] J. M. Sotoca and F. Pla, "Supervised feature selection by clustering using conditional mutual information-based distances," *Pattern Recognit.*, vol. 43, no. 6, pp. 2068–2081, 2010.

[9] M. Thoma, H. Cheng, A. Gretton, J. Han, H.-P. Kriegel, A. Smola, L. Song, P. S. Yu, X. Yan, and K. Borgwardt, "Near-optimal supervised feature selection among frequent subgraphs," in *ICDM*, 2009, pp. 1076–1087.

[10] Z. Zhao and H. Liu, "Semi-supervised feature selection via spectral analysis," in *ICDM*, 2007, pp. 641–646.

[11] Z. Xu, I. King, M. R.-T. Lyu, and R. Jin, "Discriminative semi-supervised feature selection via manifold regularization," *IEEE Trans. Neural Netw.*, vol. 21, no. 7, pp. 1033–1047, 2010.

[12] X. Kong and P. S. Yu, "Semi-supervised feature selection for graph classification," in *SIGKDD*. ACM, 2010, pp. 793–802.

[13] J. Li, K. Cheng, S. Wang, F. Morstatter, R. P. Trevino, J. Tang, and H. Liu, "Feature selection: A data perspective," *arXiv preprint arXiv:1601.07996*, 2016.

[14] X. He, D. Cai, and P. Niyogi, "Laplacian score for feature selection," in *NIPS*, 2005, pp. 507–514.

[15] Z. Zhao and H. Liu, "Spectral feature selection for supervised and unsupervised learning," in *ICML*, 2007, pp. 1151–1157.

[16] L. Du and Y.-D. Shen, "Unsupervised feature selection with adaptive structure learning," in *SIGKDD*. ACM, 2015, pp. 209–218.

[17] N. Zhou, Y. Xu, H. Cheng, J. Fang, and W. Pedrycz, "Global and local structure preserving sparse subspace learning: An iterative approach to unsupervised feature selection," *Pattern Recognit.*, vol. 53, pp. 87–101, 2016.

[18] D. Cai, C. Zhang, and X. He, "Unsupervised feature selection for multi-cluster data," in *SIGKDD*. ACM, 2010, pp. 333–342.

[19] Z. Zhao, L. Wang, and H. Liu, "Efficient spectral feature selection with minimum redundancy," in *AAAI*, Jul. 2010, pp. 673–678.

[20] Q. Gu, Z. Li, and J. Han, "Joint feature selection and subspace learning," in *IJCAI*, 2011, pp. 1294–1299.

[21] Y. Yang, H. T. Shen, Z. Ma, Z. Huang, and X. Zhou, "$\ell_{2,1}$-norm regularized discriminative feature selection for unsupervised learning," in *IJCAI*, 2011, pp. 1589–1594.

[22] C. Hou, F. Nie, D. Yi, and Y. Wu, "Feature selection via joint embedding learning and sparse regression," in *AAAI*, 2011, pp. 1324–1329.

[23] C. Hou, F. Nie, X. Li, D. Yi, and Y. Wu, "Joint embedding learning and sparse regression: A framework for unsupervised feature selection," *IEEE Trans. Cybern.*, vol. 44, no. 6, pp. 793–804, 2014.

[24] Z. Li, J. Liu, Y. Yang, X. Zhou, and H. Lu, "Clustering-guided sparse structural learning for unsupervised feature selection," *IEEE Trans. Knowl. Data Eng*, vol. 26, no. 9, pp. 2138–2150, 2014.

[25] Z. Li, J. Liu, J. Tang, and H. Lu, "Robust structured subspace learning for data representation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 10, pp. 2085–2098, 2015.

[26] P. Zhu, W. Zuo, L. Zhang, Q. Hu, and S. C. Shiu, "Unsupervised feature selection by regularized self-representation," *Pattern Recognit.*, vol. 48, no. 2, pp. 438–446, 2015.

[27] P. Zhu, W. Zhu, W. Wang, W. Zuo, and Q. Hu, "Non-convex regularized self-representation for unsupervised feature selection," *Image Vis. Comput.*, vol. 60, pp. 22–29, 2017.

[28] R. Hu, X. Zhu, D. Cheng, W. He, Y. Yan, J. Song, and S. Zhang, "Graph self-representation method for unsupervised feature selection," *Neurocomputing*, vol. 220, pp. 130–137, 2017.

[29] R. Shang, Z. Zhang, L. Jiao, C. Liu, and Y. Li, "Self-representation based dual-graph regularized feature selection clustering," *Neurocomputing*, vol. 171, pp. 1241–1253, 2016.

[30] P. Zhu, W. Zhu, Q. Hu, C. Zhang, and W. Zuo, "Subspace clustering guided unsupervised feature selection," *Pattern Recognit.*, vol. 66, pp. 364–374, 2017.

[31] J. Tang, S. Yan, R. Hong, G.-J. Qi, and T.-S. Chua, "Inferring semantic concepts from community-contributed images and noisy tags," in *MM*. ACM, 2009, pp. 223–232.

[32] Z. Li, J. Liu, X. Zhu, T. Liu, and H. Lu, "Image annotation using multi-correlation probabilistic matrix factorization," in *MM*. ACM, 2010, pp. 1187–1190.

[33] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *J. Mach. Learn. Res.*, vol. 3, pp. 1157–1182, 2003.

[34] R. Kohavi and G. H. John, "Wrappers for feature subset selection," *Artif. Intelli.*, vol. 97, no. 1, pp. 273–324, 1997.

[35] T. G. Dietterich, "Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms," *Neural Comput.*, vol. 10, no. 7, pp. 1895–1923, 1998.

[36] J. Cohen, P. Cohen, S. G. West, and L. S. Aiken, *Applied Multiple Regression/Correlation Analysis for the Behavioral Sciences.* Routledge, 2013.

[37] N. R. Draper and H. Smith, *Applied Regression Analysis.* John Wiley & Sons, 2014.

[38] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, "Imagenet Large Scale Visual Recognition Challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, 2015.

[39] A. Rozantsev, M. Salzmann, and P. Fua, "Beyond Sharing Weights for Deep Domain Adaptation," *IEEE Trans. Pattern Anal. Mach. Intell.*, DOI: 10.1109/TPAMI.2018.2814042 2018.

[40] X. Zhu and X. Wu, "Class Noise Handling for Effective Cost-Sensitive Learning by Cost-Guided Iterative Classification Filtering," *IEEE Trans. Knowl. Data Eng.*, vol. 18, no. 10, pp. 1435–1440, 2006.

[41] Q. Yang, C. Ling, X. Chai, and R. Pan, "Test-cost sensitive classification on data with missing values," *IEEE Trans. Knowl. Data Eng.*, vol. 18, no. 5, pp. 626–638, 2006.

[42] K. Nigam, A. K. McCallum, S. Thrun, and T. Mitchell, "Text Classification from Labeled and Unlabeled Documents Using EM," *Mach. Learn.*, vol. 39, no. 2–3, pp. 103–134, 2000.

[43] S. J. Pan and Q. Yang, "A Survey on Transfer Learning," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 10, pp. 1345–1359, 2010.

[44] K. Weiss, T. M. Khoshgoftaar, and D. Wang, "A Survey of Transfer Learning," *J. Big Data*, vol. 3, no. 1, pp. 1–40, 2016.

[45] R. Raina, A. Battle, H. Lee, B. Packer, and A. Y. Ng, "Self-Taught Learning: Transfer Learning from Unlabeled Data," in *ICML*, 2007, pp. 759–766.

[46] W. Dai, Q. Yang, G.-R. Xue, and Y. Yu, "Self-Taught Clustering," in *ICML*, 2008, pp. 200–207.

[47] J. Kuen, K. M. Lim, and C. P. Lee, "Self-Taught Learning of a Deep Invariant Representation for Visual Tracking via Temporal Slowness Principle," *Pattern Recognit.*, vol. 48, no. 10, pp. 2964–2982, 2015.

[48] L. Bazzani, A. Bergamo, D. Anguelov, and L. Torresani, "Self-Taught Object Localization with Deep Networks," in *WACV*, 2016, pp. 1–9.

[49] Z. Jie, Y. Wei, X. Jin, J. Feng, and W. Liu, "Deep Self-Taught Learning for Weakly Supervised Object Localization," *arXiv preprint arXiv:1704.05188*, 2017.

[50] R. Kemker and C. Kanan, "Self-taught feature learning for hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 55, no. 5, pp. 2693–2705, 2017.

[51] P. He, P. Jia, S. Qiao, and S. Duan, "Self-taught learning based on sparse autoencoder for e-nose in wound infection detection," *Sensors*, vol. 17, no. 10, p. 2279, 2017.

[52] H. Wang, F. Nie, and H. Huang, "Robust and Discriminative Self-Taught Learning," in *ICML*, 2013, pp. 298–306.

[53] S. Li, K. Li, and Y. Fu, "Self-Taught Low-Rank Coding for Visual Learning," *IEEE Trans. Neural. Netw. Learn. Syst.*, vol. 29, no. 3, pp. 645–656, 2018.

[54] S. Ravi and H. Larochelle, "Optimization as a Model for Few-Shot Learning," in *ICLR*, 2017.

[55] J. Snell, K. Swersky, and R. Zemel, "Prototypical Networks for Few-Shot Learning," in *NIPS*, 2017, pp. 4077–4087.

[56] O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra *et al.*, "Matching Networks for One Shot Learning," in *NIPS*, 2016, pp. 3630–3638.

[57] M. Zhao, S. Yue, D. Katabi, T. S. Jaakkola, and M. T. Bianchi, "Learning Sleep Stages from Radio Signals: a Conditional Adversarial Architecture," in *ICML*, 2017, pp. 4100–4109.

[58] J. Wen, J. Indulska, and M. Zhong, "Adaptive Activity Learning with Dynamically Available Context," in *PerCom*, 2016, pp. 1–11.

[59] J. Schmidhuber, "Deep Learning in Neural Networks: An Overview," *Neural Netw.*, vol. 61, pp. 85–117, 2015.

[60] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," in *Neural Comput.*, vol. 9, no. 8, 1997, pp. 1735–1780.

[61] C. Dyer, M. Ballesteros, W. Ling, A. Matthews, and N. A. Smith, "Transition-Based Dependency Parsing with Stack Long Short-Term Memory," in *ACL*, 2015.

[62] J. Wang, Y. Chen, S. Hao, X. Peng, and L. Hu, "Deep Learning for Sensor-Based Activity Recognition: A Survey," *Pattern Recognit. Lett.*, 2018.

[63] A. Bulling, U. Blanke, and B. Schiele, "A Tutorial on Human Activity Recognition Using Body-Worn Inertial Sensors," *ACM Comput. Surv.*, vol. 46, no. 3, pp. 33:1–33:33, 2014.

[64] C. Zhu and W. Sheng, "Human Daily Activity Recognition in Robot-Assisted Living Using Multi-Sensor Fusion," in *ICRA*, 2009, pp. 2154–2159.

[65] S. Lee, H. X. Le, H. Q. Ngo, H. I. Kim, M. Han, Y.-K. Lee *et al.*, "Semi-Markov Conditional Random Fields for Accelerometer-Based Activity Recognition," *Appl. Intell.*, vol. 35, no. 2, pp. 226–241, 2011.

[66] L. C. Jatoba, U. Grossmann, C. Kunze, J. Ottenbacher, and W. Stork, "Context-Aware Mobile Health Monitoring: Evaluation of Different Pattern Recognition Methods for Classification of Physical Activity," in *EMBS*, 2008, pp. 5250–5253.

[67] M. Ermes, J. Parkka, and L. Cluitmans, "Advancing from Offline to Online Activity Recognition with Wearable Sensors," in *EMBC*, 2008, pp. 4451–4454.

[68] Q. Yang, "Activity Recognition: Linking Low-Level Sensors to High-Level Intelligence," in *IJCAI*, vol. 9, 2009, pp. 20–25.

[69] K. H. Walse, R. V. Dharaskar, and V. M. Thakare, "PCA Based Optimal ANN Classifiers for Human Activity Recognition Using Mobile Sensors Data," in *ICTIS*, 2016, pp. 429–436.

[70] P. Vepakomma, D. De, S. K. Das, and S. Bhansali, "A-Wristocracy: Deep Learning on Wrist-Worn Sensing for Recognition of User Complex Activities," in *BSN*, 2015, pp. 1–6.

[71] M. Zeng, L. T. Nguyen, B. Yu, O. J. Mengshoel, J. Zhu, P. Wu, and J. Zhang, "Convolutional Neural Networks for Human Activity Recognition Using Mobile Sensors," in *MobiCASE*, 2014, pp. 197–205.

[72] S. Ha, J.-M. Yun, and S. Choi, "Multi-modal Convolutional Neural Networks for Activity Recognition," in *SMC*, 2015, pp. 3017–3022.

[73] B. Almaslukh, J. AlMuhtadi, and A. Artoli, "An Effective Deep Autoencoder Approach for Online Smartphone-Based Human Activity Recognition," *IJC-SNS*, vol. 17, no. 4, pp. 160–165, 2017.

[74] A. Wang, G. Chen, C. Shang, M. Zhang, and L. Liu, "Human Activity Recognition in a Smart Home Environment with Stacked Denoising Autoencoders," in *WAIM*, 2016, pp. 29–40.

[75] V. Radu, N. D. Lane, S. Bhattacharya, C. Mascolo, M. K. Marina, and F. Kawsar, "Towards Multimodal Deep Learning for Activity Recognition on Mobile Devices," in *UbiComp*, 2016, pp. 185–188.

[76] L. Zhang, X. Wu, and D. Luo, "Real-Time Activity Recognition on Smartphones Using Deep Neural Networks," in *UIC-ATC-ScalCom*. IEEE, 2015, pp. 1236–1242.

[77] M. Inoue, S. Inoue, and T. Nishida, "Deep Recurrent Neural Network for Mobile Human Activity Recognition with High Throughput," *Artif. Life Rob.*, vol. 23, no. 2, pp. 173–185, 2018.

[78] M. Edel and E. Köppe, "Binarized-blstm-rnn based human activity recognition," in *IPIN*, 2016, pp. 1–7.

[79] F.-F. Li, B. Fergus, and P. Perona, "A Bayesian Approach to Unsupervised One-Shot Learning of Object Categories," in *ICCV*, 2003, pp. 1134–1141.

[80] J. J. Lim, R. R. Salakhutdinov, and A. Torralba, "Transfer Learning by Borrowing Examples for Multiclass Object Detection," in *NIPS*, 2011, pp. 118–126.

[81] C. Finn, P. Abbeel, and S. Levine, "Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks," in *ICML*, 2017.

[82] T. Munkhdalai and H. Yu, "Meta Networks," in *ICML*, 2017.

[83] B. Hariharan and R. Girshick, "Low-Shot Visual Recognition by Shrinking and Hallucinating Features," in *ICCV*, 2017.

[84] A. Antoniou, A. Storkey, and H. Edwards, "Data Augmentation Generative Adversarial Networks," in *ICLR Workshops*, 2018.

[85] Y.-X. Wang, R. Girshick, M. Hebert, and B. Hariharan, "Low-Shot Learning from Imaginary Data," in *CVPR*, 2018.

[86] F. Sung, Y. Yang, L. Zhang, T. Xiang, P. H. Torr, and T. M. Hospedales, "Learning to Compare: Relation Network for Few-Shot Learning," in *CVPR*, 2018, pp. 1199–1208.

[87] V. Garcia and J. Bruna, "Few-Shot Learning with Graph Neural Networks," in *ICLR*, 2018.

[88] S. Feng and M. F. Duarte, "Graph Autoencoder-Based Unsupervised Feature Selection with Broad and Local Data Structure Preservation," *Neurocomputing*, vol. 312, pp. 310–323, 2018.

[89] M. Qian and C. Zhai, "Robust unsupervised feature selection," in *IJCAI*, 2013, pp. 1621–1627.

[90] L. Bottou and V. Vapnik, "Local learning algorithms," *Neural Comput.*, vol. 4, no. 6, pp. 888–900, 1992.

[91] D. C. Liu and J. Nocedal, "On the limited memory BFGS method for large scale optimization," *Math. Program.*, vol. 45, no. 1, pp. 501–528, 1989.

[92] M. Schmidt, "minFunc: Unconstrained differentiable multivariate optimization in matlab," Available at: http://www.cs.ubc.ca/~schmidtm/Software/minFunc.html, 2005.

[93] H. W. Kuhn, "The Hungarian method for the assignment problem," *Nav. Res. Logis.*, vol. 2, no. 1–2, pp. 83–97, 1955.

[94] A. Strehl and J. Ghosh, "Cluster ensembles – A knowledge reuse framework for combining multiple partitions," *J. Mach. Learn. Res.*, vol. 3, no. Mar, pp. 583–617, 2002.

[95] S. Feng and M. F. Duarte, "Autoencoder Based Sample Selection for Self-Taught Learning," *arXiv preprint arXiv:1808.01574*, 2018.

[96] M. Shao, D. Kit, and Y. Fu, "Generalized Transfer Subspace Learning through Low-Rank Constraint," *Int. J. Comput. Vis.*, vol. 109, no. 1–2, pp. 74–93, 2014.

[97] Q. V. Le, J. Ngiam, A. Coates, A. Lahiri, B. Prochnow, and A. Y. Ng, "On Optimization Methods for Deep Learning," in *ICML*, 2011, pp. 265–272.

[98] Y. Guo, G. Ding, J. Han, and Y. Gao, "Zero-Shot Learning with Transferred Samples," *IEEE Trans. Image Proc.*, vol. 26, no. 7, pp. 3277–3290, 2017.

[99] R. Socher, M. Ganjoo, C. D. Manning, and A. Ng, "Zero-shot Learning through Cross-Modal Transfer," in *NIPS*, 2013, pp. 935–943.

[100] P. Gehler and S. Nowozin, "On Feature Combination for Multiclass Object Classification," in *ICCV*, 2009, pp. 221–228.

[101] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[102] Y. Kim, "Convolutional Neural Networks for Sentence Classification," in *EMNLP*, 2014, pp. 1746–1751.

[103] K. J. Piczak, "ESC: Dataset for Environmental Sound Classification," in *MM*. ACM, 2015, pp. 1015–1018.

[104] B. McFee, C. Raffel, D. Liang, D. P. Ellis, M. McVicar, E. Battenberg, and O. Nieto, "librosa 0.6.1," http://dx.doi.org/10.5281/zenodo.1252297, May 2018.

[105] J. Huang, A. Gretton, K. M. Borgwardt, B. Schölkopf, and A. J. Smola, "Correcting Sample Selection Bias by Unlabeled Data," in *NIPS*, 2007, pp. 601–608.

[106] R. Aljundi, R. Emonet, D. Muselet, and M. Sebban, "Landmarks-Based Kernelized Subspace Alignment for Unsupervised Domain Adaptation," in *CVPR*, 2015, pp. 56–63.

[107] H. Lin, J. Hou, H. Yu, Z. Shen, and C. Miao, "An Agent-Based Game Platform for Exercising People's Prospective Memory," in *WI-IAT*, vol. 3, 2015, pp. 235–236.

[108] H. Xu, Z. Yang, Z. Zhou, L. Shangguan, K. Yi, and Y. Liu, "Indoor Localization via Multi-Modal Sensing on Smartphones," in *UbiComp*, 2016, pp. 208–219.

[109] D. Sánchez, M. Tentori, and J. Favela, "Activity recognition for the smart hospital," *IEEE Intell. Syst.*, vol. 23, no. 2, pp. 50–57, 2008.

[110] C. Hu, Y. Chen, X. Peng, H. Yu, C. Gao, and L. Hu, "A Novel Feature Incremental Learning Method for Sensor-Based Activity Recognition," *IEEE Trans. Knowl. Data Eng.*, 2018.

[111] R. Saeedi and A. Gebremedhin, "A Signal-Level Transfer Learning Framework for Autonomous Reconfiguration of Wearable Systems," *IEEE Trans. Mob. Comput.*, 2018.

[112] Y. Bengio, "Deep Learning of Representations: Looking Forward," in *SLSP*, 2013, pp. 1–37.

[113] Y. LeCun, Y. Bengio, and G. Hinton, "Deep Learning," *Natural*, vol. 521, no. 7553, pp. 436–444, 2015.

[114] N. Y. Hammerla, S. Halloran, and T. Ploetz, "Deep, Convolutional, and Recurrent Models for Human Activity Recognition Using Wearables," in *IJCAI*, 2016, pp. 1533–1540.

[115] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic Differentiation in PyTorch," in *NIPS Workshop*, 2017.

[116] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[117] J. Liu, S. Ji, and J. Ye, "Multi-Task Feature Learning via Efficient $\ell_{2,1}$-Norm Minimization," in *UAI*, 2009, pp. 339–348.

[118] R. Chavarriaga, H. Sagha, A. Calatroni, S. T. Digumarti, G. Tröster, J. d. R. Millán, and D. Roggen, "The Opportunity Challenge: A Benchmark Database for On-Body Sensor-Based Activity Recognition," *Pattern Recognit. Lett.*, vol. 34, no. 15, pp. 2033–2042, 2013.

[119] A. Reiss and D. Stricker, "Introducing a New Benchmarked Dataset for Activity Monitoring," in *ISWC*, 2012, pp. 108–109.

[120] S. Sukhija, "Label Space Driven Heterogeneous Transfer Learning with Web Induced Alignment," in *AAAI*, 2018, pp. 8165–8166.

[121] N. Murrugarra-Llerena and A. Kovashka, "Asking Friendly Strangers: Non-Semantic Attribute Transfer," in *AAAI*, 2018, pp. 7268–7275.